Contents

Desktop Win32 Apps
  Get Started
    Overview
    What's new
    Create your first C++ Win32 app
    Create your first Windows app using DirectX
    Using the Windows Headers
    Platform Guides
      Programming Guide for 64-bit Windows
      Compatibility cookbook for Windows
      Windows 7 guides
        Windows 7 and Windows Server 2008 R2 Application Quality Cookbook
        Windows 7 Developer Guide
        Platform Update for Windows Vista
  Design
  Develop
    Desktop App Technologies
      Overview
      Accessibility
        Overview
        What's new in Windows accessibility and automation
        Developing accessible Windows applications
        Developing accessible UI frameworks
        Developing assistive technology for Windows
        Testing for accessibility
        Legacy accessibility and automation technology
        Accessibility and automation feature reference
      Desktop App User Interface
        Overview

- Getting Started Developing User Interfaces for Windows Applications
- Internationalization for Windows Applications
- User Interaction
  - Overview
  - Direct Manipulation
  - Ink input
  - Input feedback configuration
  - Input source identification
  - Interaction context
  - Pointer device stack
  - Pointer Input Messages and Notifications
  - Radial controller input
  - Text Services Framework
  - Touch input hit testing
  - Touch input injection
  - Legacy User Interaction Features
- Windows and Messages
- Desktop Window Manager
- Dialog Boxes
- Menus and Other Resources
- Data Exchange
- High DPI
- Windows Animation Manager
- Windows Controls
- Windows Ribbon Framework

Desktop Environment
- Overview
- The Windows Shell
- Windows Property System
- Windows Search
- Window Stations and Desktops
- Legacy Windows Environment Features

Application Installation and Servicing
- Overview
- Isolated Applications and Side-by-side Assemblies
- Packaging, deployment, and query of UWP apps
- Developer licensing
- Restart Manager
- Update Orchestrator
- Windows Installer

Audio and Video
- Overview
- Core Audio APIs
- DirectShow
- Microsoft Media Foundation
- Windows Media Format 11 SDK
- Windows Media Player SDK
- Media Streaming API
- Windows Multimedia

Data Access and Storage
- Overview
- Background Intelligent Transfer Service
- Backup
- Cloud Sync Engines
- Distributed File System
- Extensible Storage Engine
- Image Mastering API
- Imaging API
- Local File Systems
- Projected File System
- Persistent Memory Programming
- Virtual Storage
- Virtual Disk Service
- Volume Shadow Copy Service

Devices
  - Overview
  - Communications Resources
  - Device access
  - Location API
  - Sensor API
  - UPnP APIs
  - Web Services on Devices
  - WIA

Windows Media Device Manager 11 SDK

Windows Portable Devices
  - Overview
  - WPD Application Programming Interface

Diagnostics
  - Overview
  - Debugging and Error Handling
    - Overview
    - Application Recovery and Restart
    - Error Handling
  - Network Diagnostics Framework
  - Network Monitor
  - Performance Counters
  - Process Snapshotting
  - Process Status API
  - System Event Notification Service
  - System Monitor
  - Tool Help Library
  - Windows Error Reporting
  - Windows Events
    - Overview
    - TraceLogging
    - Event Tracing

- Event Logging
- Windows Event Log
- Windows Event Collector

Documents and Printing

Graphics and Gaming

- Overview
- DirectX graphics and gaming
  - Overview
  - Getting started with DirectX graphics
  - Programming DirectX with COM
  - Direct2D
  - Direct3D
  - DXCore
  - DirectWrite
  - DirectXMath
  - windowsnumerics.h APIs
  - Classic DirectX Graphics
  - Tools for DirectX Graphics
  - DirectX Graphics Articles
  - XAudio2 APIs
  - XInput Game Controller APIs
- DirectComposition
- GDI+
- Windows GDI
- Monitor Configuration
- OpenGL
- Windows Imaging Component
- Windows Mixed Reality
- Windows Color System

Networking and Internet

- Overview
- Delivery Optimization

- Domain Name System
- HTTP Server API
- IP Helper
- Message Queuing
- Network Interfaces
- Network List Manager
- Network Management
- Network Share Management
- Peer-to-Peer
- Remote Procedure Call
- Routing and Remote Access Service
- Simple Network Management Protocol
- Telephony Application Programming Interfaces
- Teredo
- WebSocket Protocol Component API
- Windows Filtering Platform
- Windows Networking
- Windows Sockets 2
- Wireless Networking
  - Overview
  - Bluetooth
  - Mobile Broadband
  - Native Wifi
  - Windows Connect Now
  - Windows Connection Manager
- WebDAV
- Windows HTTP Services
- Windows Internet
- Windows Web Services

Security and Identity
- Overview
- Antimalware Scan Interface

  - Authentication
  - Authorization
  - Best Practices for the Security APIs
  - Certificate Enrollment API
  - Cryptography
  - Cryptography API: Next Generation
  - Directory, Identity, and Access Services
    - Directory Services
    - Access Management Services
  - Extensible Authentication Protocol
  - Extensible Authentication Protocol Host
  - Network Access Protection
  - Network Policy Server
  - Parental Controls
  - Rights Management
    - Overview
    - RMS Scenarios, Code and Tools
  - Security Management
  - Security Glossary
  - TPM Base Services
  - Windows Biometric Framework
- System Services
  - Overview
  - Component Object Model
  - Microsoft Interface Definition Language
  - Structured Storage
  - COM+
  - Compression API
  - Dynamic-Link Libraries
  - Interprocess Communications
  - Kernel Transaction Manager
  - Memory Management

- Power Management
- Processes and Threads
- Remote Desktop Services
- Services
- Setup API
- Synchronization
- Windows System Information

Windows Machine Learning

Developer Notes

Develop with Server Technologies

Overview

Directory, Identity, and Access Services

- Directory Services
  - Directories
  - Directory Access Technologies
- Identity Services
  - Identity Lifecycle Manager 2007
- Access Management Services
  - Active Directory Federation Services

Rights Management

- RMS Code Samples and Tools

System Administration

- Group Policy
  - Overview
  - GPMC Class Library Reference
- Microsoft Management Console (MMC)
  - Microsoft Management Console 3.0
- Windows Server Update Services
  - Windows Server Update Services 2.0 Class Library

Virtualization

WMI/MI/OMI Providers

Windows Server

- What's New For Windows Server 2012 R2 and Windows Server 2012
- Mobile Device Management Registration
- System Restore
- System Shutdown
- Task Scheduler
- Windows Deployment Services
- Windows Management Instrumentation
- Windows Management Infrastructure Managed API
- WMI Providers
  - Boot Event Collector WMI Provider
  - Hyper-V management
  - MDM Bridge WMI Provider
  - Windows System Assessment Tool
  - WMI Core Provider
- Windows Remote Management
- Windows Resource Protection

API Index

Windows Runtime C++ reference

Windows Certification

Get started with desktop Windows apps that use the Win32 API

2/18/2021 • 3 minutes to read • Edit Online

The Win32 API (also called the Windows API) is the original platform for native C/C++ Windows applications that require direct access to Windows and hardware. It provides a first-class development experience without depending on a managed runtime environment like .NET and WinRT (for UWP apps for Windows 10). This makes the Win32 API the platform of choice for applications that need the highest level of performance and direct access to system hardware.

NOTE

This documentation covers how to create desktop Windows apps with the Win32 API. The Win32 API is one of several app platforms you can use to build desktop Windows apps. For more info about other app platforms, see Choose your platform.

Get set up

Follow these instructions and start creating desktop apps for Windows 10 that use the Win32 API.

1. Download or update Visual Studio 2019. If you don't already have Visual Studio 2019, you can install the free Microsoft Visual Studio Community 2019. When you install Visual Studio, make sure to select the **Desktop development with C++** option. For download links, see our Downloads page.

   NOTE

   When you install Visual Studio, you can optionally select the **.NET desktop development** and **Universal Windows Platform development** options for access to other project types and app platforms for building desktop Windows apps.

2. If you want to build your desktop app into an MSIX package and test or debug the packaged app on your development computer, you'll need to enable Developer Mode on your computer.

NOTE

For scripts you can use to set up your development computer and install other features or packages, check out this GitHub project.

Learn how to create desktop apps using the Win32 API

If you're new to building desktop apps using the Win32 API, the following tutorials and articles will help get you started.

TOPIC	DESCRIPTION
Create your first C++ Win32 app	This tutorial teaches you how to write a Windows program in C++ using Win32 and COM APIs.

TOPIC	DESCRIPTION
Create your first app using DirectX	This basic tutorial will get you started with DirectX app development.
Programming Guide for 64-bit Windows	Describes programming for 64-bit versions of the Windows operating system.
Using the Windows Headers	Provides an overview of some of the conventions used in the Windows header files.

You can also browse the desktop app samples.

Modernize your desktop apps for Windows 10

If you have an existing desktop Win32 app, there are many features in the Universal Windows Platform (UWP) that you can use to deliver the best possible experience on Windows 10. For example, starting in Windows 10, version 1903, you can host UWP XAML controls in your desktop Win32 app using a feature called XAML Islands.

Most of these UWP features are available as modular components that you can adopt in your desktop app at your own pace without having to rewrite your entire application. You can enhance your existing desktop app by choosing which parts of Windows 10 and UWP to adopt.

For more information, see Modernize your desktop apps.

C++/WinRT

Optionally, you can configure your development computer to use C++/WinRT. C++/WinRT is an entirely standard modern C++17 language projection enables you to easily consume Windows Runtime APIs Windows Runtime (WinRT) APIs from your C++ Win32 desktop application. C++/WinRT is implemented as a header-file-based library.

To configure your project for C++/WinRT:

- For new projects, you can install the C++/WinRT Visual Studio Extension (VSIX) and use one of the C++/WinRT project templates included in that extension.
- For existing Windows desktop application projects, you can install the Microsoft.Windows.CppWinRT NuGet package in the project.

For more details about these options, see this article.

What's new for Win32 APIs in Windows 10

To learn about new Win32 APIs that have been introduced in Windows 10, see what's new.

Get started with Win32 features and technologies

Win32 APIs exist for many features and technologies in Windows 10, including core user interface and windowing APIs, audio and graphics, and networking. For guidance and code samples about using these APIs, see our features and technologies index.

Related topics

- Develop desktop apps
- Windows API reference

- Windows API index
- Windows Runtime C++ reference

What's new in Windows 10 for desktop Win32 app developers

2/18/2021 • 2 minutes to read • Edit Online

This article provides information about what's new for desktop Win32 app developers starting in Windows 10. For additional information about what's new the latest release of Windows 10, see the following resources:

- What's cool in Windows 10
- What's new in Windows 10 for UWP developers

New Win32 APIs for Windows 10

The following articles contain info about new Win32 APIs that were introduced starting in Windows 10:

- What's new in accessibility
- What's new in Direct3D 12
- What's new in DirectXMath
- What's new in Direct2D
- What's new in DirectWrite
- What's new in Background Intelligent Transfer Service (BITS)
- What's new in Power Management
- What's new in Native Wifi
- What's new in Event tracing
- What's new in Task Scheduler
- What's new in Windows Imaging Component (WIC)

Related topics

- Get started with Win32 desktop Windows apps
- What's cool in Windows 10
- Windows API reference
- Windows API index

Get Started with Win32 and C++

2/22/2020 • 2 minutes to read • Edit Online

The aim of this Get Started series is to teach you how to write a desktop program in C++ using Win32 and COM APIs.

In the first module, you'll learn step-by-step how to create and show a window. Later modules will introduce the Component Object Model (COM), graphics and text, and user input.

For this series, it is assumed that you have a good working knowledge of C++ programming. No previous experience with Windows programming is assumed. If you are new to C++, you can find learning material at the Visual C++ Developer Center. (This resource may not be available in some languages and countries.)

In this section

TOPIC	DESCRIPTION
Introduction to Windows Programming in C++	This section describes some of the basic terminology and coding conventions used in Windows programming.
Module 1. Your First Windows Program	In this module, you will create a simple Windows program that shows a blank window.
Module 2. Using COM in Your Windows Program	This module introduces the Component Object Model (COM), which underlies many of the modern Windows APIs.
Module 3. Windows Graphics	This module introduces the Windows graphics architecture, with a focus on Direct2D.
Module 4. User Input	This module describes mouse and keyboard input.
Sample Code	Contains links to download the sample code for this series.

Create your first Windows app using DirectX

2/22/2020 • 2 minutes to read • Edit Online

If you know DirectX, you can develop a DirectX app using native C++ and HLSL to take full advantage of graphics hardware.

Use this basic tutorial to get started with DirectX app development, then use the roadmap to continue exploring DirectX.

Windows desktop app with C++ and DirectX

A Windows desktop app with DirectX is an app developed using native C++ and DirectX APIs. This model is more complex than an app written in a managed framework, but it provides greater flexibility and greater access to system resources especially graphics devices. So, it is a good model for the experienced developer.

Why develop a Windows app with DirectX?

The answer is simple: you want to make a game that is graphics- or multimedia-intensive, and can use the features that many graphics devices support. This won't be easy if you are new to game development or to Windows development and C/C++, but there's some good news: DirectX 11 is the simplest and most cohesive version of Microsoft DirectX yet. It's also the most powerful and feature-rich. If your goal is to master game development and learn the most advanced rendering techniques, then DirectX can provide the opportunity for you to do that.

That said, planning your game (or interactive, real-time app) is essential. If you are new to game development, and your game doesn't have demanding graphics requirements, consider developing it with the .NET framework instead. Also, many "middleware" graphics and game development packages are available for Windows platforms, and some do not require significant programming skills.

If you are confident, or simply have a dream of making a game with high-fidelity graphics (or an app with complex graphics content), then read on!

In this section

TOPIC	DESCRIPTION
Prerequisites for developing with DirectX	When you start to develop a Windows app using DirectX, keep the prerequisites on this page in mind. This includes the technologies you need to know before you dive in.
Get started with DirectX for Windows	Creating a DirectX game for Windows is a challenge for a new developer. Here we quickly review the concepts involved and the steps you must take to begin developing a game using DirectX and C++.
Roadmap for Desktop DirectX apps	Here are key resources to help you get started with using DirectX and C++ to develop graphics-intensive Desktop apps, like games.

Using the Windows Headers

11/2/2020 • 4 minutes to read • Edit Online

The header files for the Windows API enable you to create 32- and 64-bit applications. They include declarations for both Unicode and ANSI versions of the API. For more information, see Unicode in the Windows API. They use data types that enable you to build both 32- and 64-bit versions of your application from a single source code base. For more information, see Getting Ready for 64-bit Windows. Additional features include Header Annotations and STRICT Type Checking.

- Visual C++ and the Windows Header Files
- Macros for Conditional Declarations
- Setting WINVER or _WIN32_WINNT
- Controlling Structure Packing
- Faster Builds with Smaller Header Files
- Related topics

Visual C++ and the Windows Header Files

Microsoft Visual C++ includes copies of the Windows header files that were current at the time Visual C++ was released. Therefore, if you install updated header files from an SDK, you may end up with multiple versions of the Windows header files on your computer. If you do not ensure that you are using the latest version of the SDK header files, you will receive the following error code when compiling code that uses features that were introduced after Visual C++ was released: error C2065: undeclared identifier.

Macros for Conditional Declarations

Certain functions that depend on a particular version of Windows are declared using conditional code. This enables you to use the compiler to detect whether your application uses functions that are not supported on its target version(s) of Windows. To compile an application that uses these functions, you must define the appropriate macros. Otherwise, you will receive the C2065 error message.

The Windows header files use macros to indicate which versions of Windows support many programming elements. Therefore, you must define these macros to use new functionality introduced in each major operating system release. (Individual header files may use different macros; therefore, if compilation problems occur, check the header file that contains the definition for conditional definitions.) For more information, see SdkDdkVer.h.

The following table describes the preferred macros used in the Windows header files. If you define NTDDI_VERSION, you must also define _WIN32_WINNT.

MINIMUM SYSTEM REQUIRED	VALUE FOR NTDDI_VERSION
Windows 10 1903 "19H1"	**NTDDI_WIN10_19H1** (0x0A000007)
Windows 10 1809 "Redstone 5"	**NTDDI_WIN10_RS5** (0x0A000006)
Windows 10 1803 "Redstone 4"	**NTDDI_WIN10_RS4** (0x0A000005)
Windows 10 1709 "Redstone 3"	**NTDDI_WIN10_RS3** (0x0A000004)

MINIMUM SYSTEM REQUIRED	VALUE FOR NTDDI_VERSION
Windows 10 1703 "Redstone 2"	**NTDDI_WIN10_RS2** (0x0A000003)
Windows 10 1607 "Redstone 1"	**NTDDI_WIN10_RS1** (0x0A000002)
Windows 10 1511 "Threshold 2"	**NTDDI_WIN10_TH2** (0x0A000001)
Windows 10 1507 "Threshold"	**NTDDI_WIN10** (0x0A000000)
Windows 8.1	**NTDDI_WINBLUE** (0x06030000)
Windows 8	**NTDDI_WIN8** (0x06020000)
Windows 7	**NTDDI_WIN7** (0x06010000)
Windows Server 2008	**NTDDI_WS08** (0x06000100)
Windows Vista with Service Pack 1 (SP1)	**NTDDI_VISTASP1** (0x06000100)
Windows Vista	**NTDDI_VISTA** (0x06000000)
Windows Server 2003 with Service Pack 2 (SP2)	**NTDDI_WS03SP2** (0x05020200)
Windows Server 2003 with Service Pack 1 (SP1)	**NTDDI_WS03SP1** (0x05020100)
Windows Server 2003	**NTDDI_WS03** (0x05020000)
Windows XP with Service Pack 3 (SP3)	**NTDDI_WINXPSP3** (0x05010300)
Windows XP with Service Pack 2 (SP2)	**NTDDI_WINXPSP2** (0x05010200)
Windows XP with Service Pack 1 (SP1)	**NTDDI_WINXPSP1** (0x05010100)
Windows XP	**NTDDI_WINXP** (0x05010000)

The following tables describe other macros used in the Windows header files.

MINIMUM SYSTEM REQUIRED	MINIMUM VALUE FOR _WIN32_WINNT AND WINVER
Windows 10	**_WIN32_WINNT_WIN10** (0x0A00)
Windows 8.1	**_WIN32_WINNT_WINBLUE** (0x0603)
Windows 8	**_WIN32_WINNT_WIN8** (0x0602)
Windows 7	**_WIN32_WINNT_WIN7** (0x0601)
Windows Server 2008	**_WIN32_WINNT_WS08** (0x0600)
Windows Vista	**_WIN32_WINNT_VISTA** (0x0600)

MINIMUM SYSTEM REQUIRED	MINIMUM VALUE FOR _WIN32_WINNT AND WINVER
Windows Server 2003 with SP1, Windows XP with SP2	**_WIN32_WINNT_WS03** (0x0502)
Windows Server 2003, Windows XP	**_WIN32_WINNT_WINXP** (0x0501)

MINIMUM VERSION REQUIRED	MINIMUM VALUE OF _WIN32_IE
Internet Explorer 11.0	**_WIN32_IE_IE110** (0x0A00)
Internet Explorer 10.0	**_WIN32_IE_IE100** (0x0A00)
Internet Explorer 9.0	**_WIN32_IE_IE90** (0x0900)
Internet Explorer 8.0	**_WIN32_IE_IE80** (0x0800)
Internet Explorer 7.0	**_WIN32_IE_IE70** (0x0700)
Internet Explorer 6.0 SP2	**_WIN32_IE_IE60SP2** (0x0603)
Internet Explorer 6.0 SP1	**_WIN32_IE_IE60SP1** (0x0601)
Internet Explorer 6.0	**_WIN32_IE_IE60** (0x0600)
Internet Explorer 5.5	**_WIN32_IE_IE55** (0x0550)
Internet Explorer 5.01	**_WIN32_IE_IE501** (0x0501)
Internet Explorer 5.0, 5.0a, 5.0b	**_WIN32_IE_IE50** (0x0500)

Setting WINVER or _WIN32_WINNT

You can define these symbols by using the #define statement in each source file, or by specifying the /D compiler option supported by Visual C++.

For example, to set WINVER in your source file, use the following statement:

```
#define WINVER 0x0502
```

To set _WIN32_WINNT in your source file, use the following statement:

```
#define _WIN32_WINNT 0x0502
```

To set _WIN32_WINNT using the /D compiler option, use the following command:

cl -c /D_WIN32_WINNT=0x0502 *source*.cpp

For information on using the /D compiler option, see /D (preprocessor definitions).

Note that some features introduced in the latest version of Windows may be added to a service pack for a previous version of Windows. Therefore, to target a service pack, you may need to define _WIN32_WINNT with the value for the next major operating system release. For example, the **GetDllDirectory** function was introduced in Windows Server 2003 and is conditionally defined if _WIN32_WINNT is 0x0502 or greater. This function was also added to Windows XP with SP1. Therefore, if you were to define _WIN32_WINNT as 0x0501 to target Windows XP, you would miss features that are defined in Windows XP with SP1.

Controlling Structure Packing

Projects should be compiled to use the default structure packing, which is currently 8 bytes because the largest integral type is 8 bytes. Doing so ensures that all structure types within the header files are compiled into the application with the same alignment the Windows API expects. It also ensures that structures with 8-byte values are properly aligned and will not cause alignment faults on processors that enforce data alignment.

For more information, see /Zp (struct member alignment) or pack.

Faster Builds with Smaller Header Files

You can reduce the size of the Windows header files by excluding some of the less common API declarations as follows:

- Define WIN32_LEAN_AND_MEAN to exclude APIs such as Cryptography, DDE, RPC, Shell, and Windows Sockets.

  ```
  #define WIN32_LEAN_AND_MEAN
  ```

- Define one or more of the NO*api* symbols to exclude the API. For example, NOCOMM excludes the serial communication API. For a list of support NO*api* symbols, see Windows.h.

  ```
  #define NOCOMM
  ```

Related topics

Windows SDK download site

Programming Guide for 64-bit Windows

2/22/2020 • 2 minutes to read • Edit Online

Microsoft has released 64-bit versions of the Windows operating system. 64-bit Windows was designed with compatibility in mind. Developers can ensure that their existing 32-bit applications run well under 64-bit Windows or take advantage of the benefits of 64-bit Windows by migrating their applications.

Benefits of 64-bit Windows

A 64-bit operating system supports far more physical memory than a 32-bit operating system. For example, most 32-bit Windows systems support a maximum of 4 gigabytes of physical memory, with up to 3 gigabytes of address space for each process, while 64-bit Windows supports up to 2 terabytes of physical memory with 8 terabytes of address space for each process. The increased physical memory includes the following benefits for applications:

- Each application can support more users. All or part of each application must be replicated for each user, which requires additional memory.
- Each application has better performance. Increased physical memory allows more applications to run simultaneously and remain completely resident in the system's main memory. This reduces or eliminates the performance penalty of swapping pages to and from disk.
- Each application has more memory for data storage and manipulation. Databases can store more of their data in the physical memory of the system. Data access is faster because disk reads are not necessary.
- Applications can manipulate large amounts of data easily and more reliably. Video composition for motion picture work requires 64-bit Windows for this reason. Modeling for scientific and financial applications benefits greatly from memory-resident data structures that are not possible on 32-bit Windows.

There are also important benefits for businesses:

- Increased productivity. Knowledge workers can spend their time thinking and producing, rather than waiting for the software to finish its tasks.
- Lower cost of ownership. Each server can support larger numbers of users and applications, so your business will require fewer servers. This translates directly into less management overhead—one of the highest costs in any computing environment.
- New application opportunities. New applications can be designed without the barriers imposed by 32-bit Windows. New graphics applications will make work easier and more enjoyable. Data-intensive tasks that are impossible today can be done with 64-bit Windows.

In this Section

- Getting Ready for 64-bit Windows
- Designing 64-bit Compatible Interfaces
- Running 32-bit Applications
- Migration Tips

Windows and Windows Server compatibility cookbook: Windows 10, Windows 8, Windows 8.1, Windows Server 2012, and Windows Server 2012 R2

2/22/2020 • 2 minutes to read • Edit Online

Purpose

In the *Cookbook*, we provide info about changes to and new features of the Windows 10, Windows 8, Windows 8.1, Windows Server 2012, and Windows Server 2012 R2 operating systems. We also provide guidelines for you to verify the compatibility of your existing and planned programs with the new operating systems. We assume that you are familiar with previous versions of Windows.

The content applies to:

- Windows 10
- Windows 8.1
- Windows Server 2012 R2
- Windows 8
- Windows Server 2012
- Windows 7
- Windows Server 2008 R2

Windows 7

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Windows 7 and Windows Server 2008 R2 Application Quality Cookbook
- Windows 7 Developer Guide
- Hilo: Developing C++ Applications for Windows 7
- Platform Update for Windows Vista

Windows 7 and Windows Server 2008 R2 Application Quality Cookbook

2/18/2021 • 2 minutes to read • Edit Online

The *Windows 7 and Windows Server 2008 R2 Application Quality Cookbook* familiarizes application developers with how to verify the compatibility of their applications with the new operating system and provides an overview of the few known application compatibility issues in Windows 7 and Windows Server 2008 R2. It also points out differences in performance, reliability, and usability, and provides links to detailed white papers and other developer guidance.

Overview

Windows 7 and Windows Server 2008 R2 introduce the latest operating system technology and software development platform for use by application developers and enterprises worldwide. As part of further enhancing the security, reliability, performance, and user experience of Windows, many new features have been introduced, existing features have been improved, and some features have been removed.

While Windows 7 and Windows Server 2008 R2 are highly compatible with most of their respective applications written for Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2 and their service packs, some compatibility breaks are inevitable due to innovations, security tightening, and increased reliability. Overall, the compatibility of Windows 7 and Windows Server 2008 R2 with existing applications is high.

This document builds on the concepts embodied in the Windows Vista and Windows Server 2008 Application Compatibility Cookbook. (This resource may not be available in some languages and countries/regions.) Like it, this document provides you with the means to become familiar with how to verify the compatibility of your applications with the new operating system and provides an overview of the few known application incompatibility issues in Windows 7 and Windows Server 2008 R2. But more than that, it also points out differences in performance, reliability, and usability, and provides links to detailed white papers and other developer guidance.

In addition, Microsoft is investing in several new and enhanced features and tools to enable you to build higher quality applications and to troubleshoot when applications do not function properly on Windows 7 and Windows Server 2008 R2. Included among these tools are instructions for how to qualify your client and server applications for participating in the Windows Logo Program.

This *Cookbook* contains more than three dozen topics divided into three major sections:

- Compatibility (both General and Server-specific)
- New Features and Enhancements
- Tools, Best Practices, and Guidance

We invite you to check out these topics to learn how to optimize your application to take advantage of what Windows 7 has to offer.

We take your suggestions for improvements to this document seriously.

Windows 7 Developer Guide

11/2/2020 • 2 minutes to read • Edit Online

Building applications that are easy to use, visually appealing, and offer high performance is a challenge that developers face every day. Innovative applications can greatly improve the user experience, empowering companies to differentiate their services and solutions. However, developers are increasingly asked to do more in less time, while also optimizing the power and performance requirements of their applications.

The Windows 7 platform makes it easy for developers to create engaging, user-friendly applications by providing familiar tools and rich development features that allow them to take advantage of the latest PC capabilities.

For a downloadable version of the Windows 7 Developer Guide, visit Code Gallery on MSDN.

In the Windows 7 Developer Guide:

- Solid Foundation
- Richer Application Experiences
- The Best of Windows and the Web

Platform Update for Windows Vista

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The Platform Update for Windows Vista and the Platform Update for Windows Server 2008 are end-user operating system updates that support the use of selected Windows 7 technologies on previous versions of the Windows operating system. The updates include a set of runtime libraries that enable application developers to target current releases, Windows 7 and Windows Server 2008 R2 as well as previous versions, Windows Vista and Windows Server 2008.

Developer audience

The API supported by the Platform Update for Windows Vista and the Platform Update for Windows Server 2008 are designed for use by experienced C/C++ developers.

Recommended proficiencies:

- COM programming
- Windows API programming

Run-time requirements

For Windows Vista, end-users obtain the Platform Update for Windows Vista from Windows Update. For Windows Server 2008, end-users obtain the Platform Update for Windows Server 2008 from Windows Update.

In this section

TOPIC	DESCRIPTION
About Platform Update for Windows Vista	Provides an overview of Platform Update for Windows Vista and Platform Update for Windows Server 2008 and their features.
Developing Applications for Previous Versions of Windows	Explains what to do to develop applications that run on previous versions of Windows and take advantage of the API that are supported with the Platform Update for Windows Vista and the Platform Update for Windows Server 2008.

Additional resources

Knowledge Base article about the Platform Update for Windows Vista (KB 971644)

Guidelines

2/18/2021 • 2 minutes to read • Edit Online

NOTE

This design guide was created for Windows 7 and has not been updated for newer versions of Windows. Much of the guidance still applies in principle, but the presentation and examples do not reflect our current design guidance.

These sections comprise the detailed user experience guidelines for Windows-based desktop applications.

In this section

TOPIC	DESCRIPTION
Controls	Controls are UI elements that your users interact with on your app's main window area. See visual examples of controls in Windows-based, desktop apps and get links to guidelines for each control.
Commands	Commands are actions users can take while using your app. Learn the guidelines for adding commands to your app's menus, ribbons, and toolbars.
Text	Text includes any text users can see in your app. Review these guidelines on the use of UI text, style, and tone for your apps.
Messages	Messages are any kind of message users need or want to see as they use your app. Learn how to present errors, warning, confirmations, and notifications in your app.
Interaction	Interaction is the variety of ways users interact with your app, including touch, keyboard, mouse, and so on. Use these guidelines to create intuitive and distinctive experiences that are optimized for touch but work consistently across input devices.
Windows	Windows are the main "canvases" or UI surfaces of your desktop app, including the main windows itself and pop-ups, dialogs, and wizards. Follow these guidelines when deciding which surface to use and how best to use them.
Visuals	Visuals include the visual elements other than the controls. These guidelines help you make decisions about layout, fonts, color, icons, and so on in your app.
Experiences	Experiences are the common experiences and use cases for all apps, like set up, first run, and printing. Learn about the best practices for creating these experiences and communicating your app brand.

TOPIC	DESCRIPTION
Windows Environment	The Windows environment is the onscreen work area provided by Windows, analogous to a physical desktop, and the operating system's core extension points. Learn how to leverage the desktop, taskbar, notification area, control panels, help, and user account control for your app.

Desktop app technologies

2/18/2021 • 2 minutes to read • Edit Online

This section provides in-depth guidance and code examples about Windows features that are available to desktop applications by using the Win32 API.

In this section

TOPIC	DESCRIPTION	
Accessibility	Provides guidance for Windows developers designing accessible applications, assistive technology developers building tools such as screen readers and magnifiers, and software test engineers creating automated scripts for testing Windows applications.	
Desktop user interface	Provides information that enables you to develop graphical user interfaces for your apps, including features such as windows and messages, resources, and controls.	
Desktop environment	Provides guidance for integrating and extending the desktop user-facing features of Windows, including the Taskbar, the desktop, and File Explorer.	
Application installation and servicing	Provides information about using APIs and services provided by Windows to install, manage, and service your desktop apps.	
Audio and video	Provides guidance about using audio and video features provided by Windows.	
Data access and storage	Provides information about data access and storage features you can use in your desktop applications, including file system management and cloud sync engines.	
Devices	Describes APIs for interacting with devices and sensors.	
Diagnostics	Provides guidance about debugging and error handling, performance profiling, network monitoring, and other diagnostics features.	

TOPIC	DESCRIPTION	
Documents and printing	Describes the documents and printing features of Windows that enable applications to save, view, and print.	
Graphics and gaming	Provides information about graphics and gaming features of Windows, including DirectX and digital imaging.	
Networking and Internet	Provides guidance about the networking and Internet-related features of Windows, including network management, HTTP APIs, and Remote Procedure Call (RPC).	
Security and identity	Provides information about authentication, authorization, cryptography, and other security features of Windows.	
System services	Provides guidance about fundamental OS features such as process and threads, services, dynamic-link libraries, COM, the registry, and more.	information.

Windows accessibility

2/18/2021 • 2 minutes to read • Edit Online

This section includes documentation for Windows developers designing accessible applications, assistive technology developers building tools such as screen readers and magnifiers, and software test engineers creating automated scripts for testing Windows applications.

Use the resources provided here to build accessible Windows applications that can be used by as many people as possible, including those with disabilities, personal preferences, and specific work styles.

In this section

TOPIC	DESCRIPTION
New Windows accessibility features, tools, documentation, and samples for developers	The Windows platform is constantly being updated with new accessibility and automation features for developers.
Developing accessible applications for Windows	Each Windows application framework supports numerous accessibility features that make it easier for users with disabilities, personal preferences, or specific work styles to successfully use any Windows device they choose.
Developing accessible UI frameworks for Windows	To be usable by as many people as possible, UI frameworks built for the Windows platform should support accessibility features that make it easier for those with disabilities, personal preferences, or specific work styles to use their Windows devices successfully.
Developing assistive technology for Windows	Third-party assistive technology products—such as screen readers, magnifiers and speciality hardware—are essential tools for many users of Windows (and other Microsoft products).
Testing with UI Automation and other tools	Testing the accessibility of your Windows applications, assistive technology tools, and UI frameworks is a critical step in ensuring successful user experiences for people with disabilities and other limitations, as well as users who prefer keyboard interactions.
Legacy accessibility and automation technology - MSAA to UI Automation	Windows accessibility and automation consist of two technologies—Microsoft Active Accessibility (MSAA) and Microsoft UI Automation. MSAA is a legacy technology introduced with Windows 95, while UI Automation is a newer, more capable technology that addresses the limitations of MSAA.
Windows accessibility and automation reference	The Desktop Win32 APIs provide accessibility and automation features to help developers build applications and UI frameworks, assistive technology vendors build tools, testers ensure quality implementations, and people with disabilities use their computers and devices.

New Windows accessibility features, tools, documentation, and samples for developers

2/18/2021 • 2 minutes to read • Edit Online

The Windows platform is constantly being updated with new accessibility and automation features for developers.

Install the tools and SDK on Windows 10 and you're ready to either create a new Universal Windows app or explore how you can use your existing app code on Windows.

For more info on the latest tools available for testing the accessibility implementation of your Windows and web applications, see Testing for accessibility.

What's new for Windows 10 May 2019 Update (Version 1903)

The following features, tools, developer guidance, samples, and videos have been made available for the Windows 10 May 2019 Update.

RELEASE	PLATFORM	FEATURE	DESCRIPTION	LINK
1903	Win32	UI Automation supports IAccessible2	UI Automation clients can now seamlessly access information from IAccessible2 providers such as the Chrome browser.	n/a
1903	UWP	Scrollbar visibility	Scroll bars automatically hidden when not being interacted with.	UISettings.AutoHideScrollBars Property
1903	Win32	UI Automation supports structured markup	Intended, but not limited to, MathML parsing.	n/a

What's new for Windows 10 October 2018 Update (Version 1809)

The following features, tools, developer guidance, samples, and videos have been made available for the Windows 10 October 2018 Update.

RELEASE	PLATFORM	FEATURE	DESCRIPTION	LINK

RELEASE	PLATFORM	FEATURE	DESCRIPTION	LINK
1809	Win32	UI Automation supports Active Text Position Change events	UI Automation providers can explicitly set a starting position within a text range. Assistive technology (AT) clients can then convey this position to a user. For example, if a user clicks a link to an anchor on the same page (a bookmark link), the new location is provided to the AT.	IUIAutomation6 interface
1809	Win32	UI Automation supports event coalescing	Improves performance by attempting to detect, filter, and ignore duplicate sequential MSAA and UIA TextChanged events raised by some providers.	IUIAutomation6 interface
1809	Win32	UI Automation supports connection recovery behavior	Messages from a UI Automation client to a provider are suspended (for two seconds, by default) if the provider is non-responsive. This reduces the frequency of extended timeouts, and minimizes flooding a non-responsive provider with events and requests.	IUIAutomation6 interface
1809	UWP	Enhanced screen reader services	AT clients know the audible screen reading location (control or content) and the reading mode.	ScreenReaderService Class
1809	Win32, UWP	UI Automation supports dialog windows	UI Automation providers can mark UIA elements specifically as dialog windows. ATs often present dialogs differently to users.	IUIAutomationElement9 ContentDialog Class

RELEASE	PLATFORM	FEATURE	DESCRIPTION	LINK
1809	UWP	Text scaling	Scales text size in applications and web pages.	UISettings.TextScaleFactorChanged Event Text scaling

Developing accessible applications for Windows

2/18/2021 • 2 minutes to read • Edit Online

Each Windows application framework supports numerous accessibility features that make it easier for users with disabilities, personal preferences, or specific work styles to successfully use any Windows device they choose.

This overview describes the accessibility features supported in each framework, and how you can incorporate them as you build your application.

In this section

- Web applications
- Win32 apps
- UWP apps
- WinForms apps
- WPF apps

Developing accessible UI frameworks for Windows

2/18/2021 • 2 minutes to read • Edit Online

To be usable by as many people as possible, UI frameworks built for the Windows platform should support accessibility features that make it easier for those with disabilities, personal preferences, or specific work styles to use their Windows devices successfully.

This overview describes how to build a UI framework that supports features such as programmatic access and automation, keyboard navigation and commanding, color and theme options, and personalization through user settings.

In this section

- UI Automation for Win32
- Keyboard accessibility
- Respecting High Contrast
- User settings
- User settings blog post

Samples

- UI Automation WPF Framework Implementation sample
- UI contrast and settings sample

Developing assistive technology for Windows

2/18/2021 • 2 minutes to read • Edit Online

Third-party assistive technology products—such as screen readers, magnifiers and speciality hardware—are essential tools for many users of Windows (and other Microsoft products).

This overview describes how to build assistive technology tools that are compatible with Microsoft products for people with vision, learning, dexterity/mobility, and language/communication disabilities or limitations.

In this section

- UI Automation for Win32
- UI Automation for .NET Framework
- Security Considerations for Assistive Technologies
- Ease of Access Assistive Technology Registration
- Magnification API

Testing for accessibility

2/18/2021 • 3 minutes to read • Edit Online

Programmatic access and keyboard access are critical requirements for supporting accessibility in your application. Testing the accessibility of your Windows applications, assistive technology (AT) tools, and UI frameworks is crucial to ensure a successful user experience for people with various disabilities and limitations (including vision, learning, dexterity/mobility, and language/communication disabilities), or those who simply prefer using a keyboard.

Without adequate access through AT such as screen-readers and on-screen keyboards, users with vision, learning, dexterity/mobility, and language/communication disabilities or limitations (and users who just prefer using the keyboard) would be unable to use your application.

This section describes the various tools that can be used to test the accessibility implementation of both Windows and web applications.

NOTE

It is also important to do manual testing to verify keyboard access to your application.

Tools

Accessibility Insights - Helps developers find and fix accessibility issues in both websites and Windows applications.

- Accessibility Insights for Web is an extension for Chrome and Microsoft Edge Insider that helps developers find and fix accessibility issues in web apps and sites. It supports two primary scenarios:
  - **FastPass** - a lightweight, two-step process that helps developers identify common, high-impact accessibility issues in less than five minutes.
  - **Assessment** - lets anyone verify that a web site is 100% compliant with accessibility standards and guidelines. Accessibility Insights also lets you review UI Automation elements, properties, control patterns, and events (similar to the Inspect and AccEvent legacy tools described in the following section).
- Accessibility Insights for Windows helps developers find and fix accessibility issues in Windows apps. The tool supports three primary scenarios:
  - **Live Inspect** lets developers verify that an element in an app has the right UI Automation properties simply by hovering over the element or setting keyboard focus on it.
  - **FastPass** - a lightweight, two-step process that helps developers identify common, high-impact accessibility issues in less than five minutes.
  - **Troubleshooting** allows you to diagnose and fix specific accessibility issues.

Legacy testing tools

The following tools are still available in the Windows SDK and are documented here for continued support, but we recommend transitioning to Accessibility Insights.

- Inspect: Lets you view the accessibility data of any UI element. It is especially useful for ensuring properties and control patterns are set correctly when extending a common control or creating a custom control.
- Accessible Event Watcher (AccEvent): Examines accessibility data to validate application UI elements and

ensure UI elements raise proper Microsoft Active Accessibility and UI Automation events on UI events. AccEvent is typically used to debug issues and to validate that custom and extended controls are working correctly.

- AccScope: Enables visual evaluation of an application's accessibility during the early design and development phases. AccScope helps visualize how a screen reader uses UI Automation information provided by an app, and shows where adding information or support to your application can improve its accessibility.
- UI Accessibility Checker: Verifies that key UI accessibility requirements in an application are achieved. AccChecker includes verification checks for UI Automation, Microsoft Active Accessibility, and Accessible Rich Internet Applications (ARIA). It can provide a static check for errors such as missing names, tree issues and more. It helps verify programmatic access and includes advanced features for automating accessibility testing.
- UI Automation Verify: A framework for manual and automated testing of the UI Automation implementation in a control or application (results can be logged). You can integrate your application into the test code and conduct regular, automated testing or spot checks of your UI Automation scenarios. This tool is useful for verifying that changes to applications with established features do not have new issues or regressions in areas beyond the new features.

Legacy accessibility and automation technology - MSAA to UI Automation

2/18/2021 • 2 minutes to read • Edit Online

Windows accessibility and automation consist of two technologies—Microsoft Active Accessibility (MSAA) and Microsoft UI Automation. MSAA is a legacy technology introduced with Windows 95, while UI Automation is a newer, more capable technology that addresses the limitations of MSAA.

This section describes the main differences between MSAA and UI Automation (and how to transition to UI Automation).

In this section

- Microsoft Active Accessibility and UI Automation Compared
- Microsoft Active Accessibility

Windows accessibility and automation reference

2/18/2021 • 2 minutes to read • Edit Online

Windows Accessibility supports two categories of features to help Windows developers design accessible applications, assistive technology developers build tools such as screen readers and magnifiers, and software test engineers create automated scripts for testing Windows applications: User settings and a comprehensive set of Win32 APIs.

For more information, see Windows Accessibility features.

Desktop App User Interface

2/18/2021 • 3 minutes to read • Edit Online

This section provides information that enables you to develop graphical user interfaces for your apps.

In this section

TOPIC	DESCRIPTION
Getting Started Developing User Interfaces for Windows Applications	Provides general guidance to developers who are designing, implementing, and testing the user interface of a Windows application.
Windows User Experience Interaction Guidelines	Provides user experience guidelines for Windows-based desktop applications.
Internationalization for Windows Applications	Describes the technologies that enable you to support the many cultures and written languages of the international marketplace in your Windows-based applications.
Accessibility	Describes accessibility features that make it easier for persons with disabilities to use computers.
User Interaction	Describes features that enable the user to interact with an application, through devices such as the keyboard, mouse, and touch screens.
Windows and Messages	Describes the elements of an application with a Windows-based graphical user interface.
Desktop Window Manager	Desktop Window Manager (DWM) enables visual effects on the desktop as well as various features such as glass window frames, 3-D window transition animations, Windows Flip and Windows Flip3D, and high resolution support.
Dialog Boxes	A dialog box is a temporary window an application creates to retrieve user input. An application typically uses dialog boxes to prompt the user for additional information for menu items. A dialog box usually contains one or more controls (child windows) with which the user enters text, chooses options, or directs the action.
Menus and Other Resources	A resource is binary data that you can add to the executable file of a Windows-based application. A resource can be either standard or defined. The data in a standard resource describes an icon, cursor, menu, dialog box, bitmap, enhanced metafile, font, accelerator table, message-table entry, string-table entry, or version information. An application-defined resource, also called a custom resource, contains any data required by a specific application.
Data Exchange	Describes basic methods of exchanging data, such as the clipboard and Dynamic Data Exchange.

TOPIC	DESCRIPTION
High DPI	Writing a DPI-aware application is the key to making a UI look consistently good across a wide variety of high-DPI display settings. An application that is not DPI-aware but is running on a high-DPI display setting can suffer from many visual artifacts, including incorrect scaling of UI elements, clipped text, and blurry images. By adding support in your application for DPI awareness, you guarantee that the presentation of your application's UI is more predictable, making it more visually appealing to users.
Windows Animation Manager	The Windows Animation Manager (Windows Animation) enables rich animation of user interface elements. It is designed to simplify the process of adding animation to an application's user interface and to enable developers to implement animations that are smooth, natural, and interactive.
Windows Controls	A control is a child window that an application uses in conjunction with another window to enable user interaction. Controls provide the user a way to view and edit text, choose options, choose commands, initiate actions, and view status.
Windows Ribbon Framework	The Windows Ribbon framework is a rich command presentation system that provides a modern alternative to the layered menus, toolbars, and task panes of traditional Windows applications. The Ribbon framework is composed of a ribbon command bar that exposes the major features of an application through a series of tabs at the top of an application window, and a context menu system.
Tiles, badges, and notifications for Classic desktop applications	Describes how to respond to toast notifications that appear in the action center. These toasts can be used to simply activate your application, or they can be used to gather information from the user and alter the launch protocol based on that information.
Title Callable UI	Do not use. This API set is only supported for Xbox developers.

Related topics

Windows Environment Development

Getting Started Developing User Interfaces for Windows Applications

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The following sections offer general guidance to developers who are designing, implementing, and testing the user interface of a Windows application.

In addition to basic user interface design principles, numerous recommendations and suggestions are provided that will help developers provide a user experience that is as simple, efficient, and enjoyable as possible.

NOTE

These guidelines are not intended to be comprehensive and are subject to the specific scope and functionality of an application. For more comprehensive guidelines, see the Windows User Experience Interaction Guidelines.

In this section

TOPIC	DESCRIPTION
Overview of the User Interface Development Process	This section outlines the three phases of user interface design and introduces the tasks that are typically associated with each phase.
Designing a User Interface	This section describes in detail some of the tasks associated with designing a UI for a Windows application.
Implementing a User Interface	This section describes some of the tasks associated with implementing a user interface for a Windows application.
Testing a User Interface	This section describes in detail some of the tasks associated with testing a UI for a Windows application.
Security Considerations: Windows User Interface	This topic provides information about security considerations in the Windows User Interface.
Other Resources	This section contains a list of recommended books and resources related to user interface design. (These books and resources may not be available in some languages and countries.)

Internationalization for Windows Applications

2/18/2021 • 4 minutes to read • Edit Online

(Formerly titled "International Support")

This section describes the technologies in Windows that enable you to support the many cultures and written languages of the international marketplace in your C or C++ based Microsoft Win32 application.

Windows has become an essential platform for customers worldwide. International users expect solutions that are adapted to their languages and regions around the world. In this section, you will find the information you need to develop multilanguage, multicultural, and multi-site solutions. The international support built into Windows empowers you to implement many scenarios with less engineering overhead than ever before.

The development of world-ready applications requires the use of many services and tools. Windows contains features that enable you to develop solutions that:

- Support the different language-specific and locale-specific needs of users around the world (including specialized text support, sorting behavior, date and time formatting, and keyboard layouts). (For more information, see National Language Support Knowledge Center.)
- Are globalized (can be deployed worldwide from a single binary image) and can be localized (able to be adapted for specific local markets). (For more information, see Multilingual User Interface.)
- Display international fonts and text, and allow users to specify the font they want. (For more information, see Script and Font Support in Windows.)
- Permit the user to enter complex characters and symbols with a standard keyboard.
- Provide support for many different written languages through Unicode and traditional character sets.
- Discover the language input by a user, and tailor the user experience provided by your application. (For more information, see Writing World-Ready Applications in Windows: Extended Linguistic Services in Windows.)

In this Section

The following international support technologies are documented in this section. They are listed with some key scenarios for which they can be used.

- Getting Started with International Windows Development

  Describes how to get started creating world-ready applications, and provides a tutorial illustrating a common task in writing global software.

  Common Scenarios:

  - Determine a path to take to learn how to develop international software.
  - Discover the internationalization technologies available in the Microsoft Windows Software Development Kit (SDK).
  - Follow a tutorial that takes an existing monolingual application and adds support for additional languages.
- Globalization Services

  Describes Extended Linguistic Services (ELS), which enable you to discover the language in which text and user input is written, and National Language Support (NLS), which enables an application to use locale information to display culture-sensitive information (such as time, dates, and currency) and properly sort strings.

  Common Scenarios:

  - Discover the language of the user's input, so that help content can be displayed in an understandable language.
  - Discover the script used in text that is to be displayed. If it is Simplified or Traditional Chinese, offer the user the option to have the text transliterated from one to the other.
  - Permit the user to select a locale (a collection of language-related user preference information).
  - Display times, dates, calendar information, currency, and many other culture-dependent objects in appropriate languages and formats.
  - Sort strings into the order expected by the user of a given locale.

- Input Method Manager

  Describes the technology used by an application to communicate with an input method editor (IME). The IME allows computer users to enter complex characters and symbols by using a standard keyboard.

  Common Scenario:

  - Permit the user to use a standard keyboard to enter Japanese kanji characters.

- International Fonts and Text Display

  Describes the support provided by the Windows platform for international fonts, international text, and fine typography.

  Common Scenarios:

  - Allow the user to select international fonts based on character set.
  - Display international text.
  - Process complex scripts, including bidirectional rendering, contextual shaping, and ligatures (Uniscribe).
  - Allow a high degree of control for fine typography (Uniscribe).

- Multilingual User Interface

  Describes how applications can separate language-dependent resources from language-neutral code for supported user interface languages.

  Common Scenarios:

  - Create regional or worldwide single deployment images of an application.
  - Localize a solution by updating application resources with no change to application source code.
  - Permit users to switch from one UI language to another at run time.

- Unicode and Character Sets

  Describes how applications can take advantage of Unicode, the worldwide character-encoding standard that uses 16-bit code values to represent all the characters used in modern computing, including technical symbols and special characters used in publishing.

  Common Scenarios:

  - Support the many different languages of the international marketplace through Unicode.
  - Convert Unicode characters to and from other character sets, when necessary.

- Security Considerations: International Features

  Provides information about security considerations related to international development support features.

  The security information pertains to all scenarios.

Related International Technologies

International development support is also available for applications written in managed code. If you are developing for the .NET Framework, you will need some or all of these:

- The System.Globalization Namespace contains classes that define culture-related information and provide advanced globalization functions.
- The System.Text Namespace contains classes that represent character encodings, convert blocks of characters, and manipulate and format String objects.

User Interaction

2/18/2021 • 2 minutes to read • Edit Online

In this section

TOPIC	DESCRIPTION
Direct Manipulation	The Direct Manipulation APIs are used to process touch input on a region or object, generate output to be used by a composition engine for applying transforms to the related rendering surface, and optimize response and reduce latency through predictive output based on the rendering time of the compositor.
Ink input	Learn about the various Desktop inking APIs for Windows apps.
Input Feedback Configuration	The Input Feedback Configuration APIs enable you to configure visual feedback for user interactions based on the Windows application or UI framework you are developing.
Input Source Identification	The Input Source Identification APIs enable Windows applications to accurately and reliably detect an input source, identify the input type, and provide a user experience that's most appropriate for the input type.
Interaction Context	The Interaction Context enable Windows applications and UI frameworks to support multiple, concurrent interactions by providing gesture detection and manipulation processing.
Pointer Device Input Stack	The topics in this section provide an overview of the Windows Pointer Device Input Stack Reference APIs. The Pointer Device Input Stack provides information about the input devices connected to the system.
Pointer Input Messages and Notifications	The Pointer Input Message APIs expose pointer-related input messages, notifications, and associated functionality in Windows applications.
Radial controller input	Learn about the various Desktop APIs for Windows Wheel accessories such as the Surface Dial (available for purchase at the Microsoft Store).
Touch Hit Testing	Use the Touch Hit Testing APIs to programmatically identify an input target by determining whether a geometry or point falls within the content area of a UI element.
Touch Injection	The Touch Injection APIs enable Windows developers to programmatically simulate touch input.
Legacy User Interaction Features	User interaction features for Windows 7 and earlier.

Direct Manipulation

2/18/2021 • 3 minutes to read • Edit Online

The Direct Manipulation APIs let you create great pan, zoom, and drag user experiences. To do this, it processes touch input on a region or object, generates output transforms, and applies the transforms to UI elements. You can use Direct Manipulation to optimize responsiveness and reduce latency through off-thread input processing, optional off-thread input hit testing, and input/output prediction.

Any application that uses Direct Manipulation to process touch interactions displays the fluid Windows 8 animations and interaction feedback behaviors that conform to the Guidelines for common user interactions.

Developer Audience

The Direct Manipulation API is for experienced developers who know C/C++, have a solid understanding of the Component Object Model (COM), and are familiar with Windows programming concepts.

Run-time requirements

Direct Manipulation was introduced in Windows 8. It is included in both 32-bit and 64-bit versions.

Why use DirectManipulation

Handles interactions in a straightforward and consistent manner

Direct Manipulation works by pre-declaring the behaviors and interactions for a region or object. For example, a web page is often configured for pan and zoom. At runtime, input is then associated with this region/object through a simple API call. From this point forward Direct Manipulation does all the heavy lifting of processing the input, applying constraints and personality, and generating the output transforms.

Build responsive touch applications

To optimize responsiveness and minimize latency, Direct Manipulation processing occurs on a separate, independent thread from the UI thread. As a result, output transforms can run in parallel to activity on the UI thread. The UI thread activity may include application logic, rendering, layout, and anything else that consumes cycles on the processor.

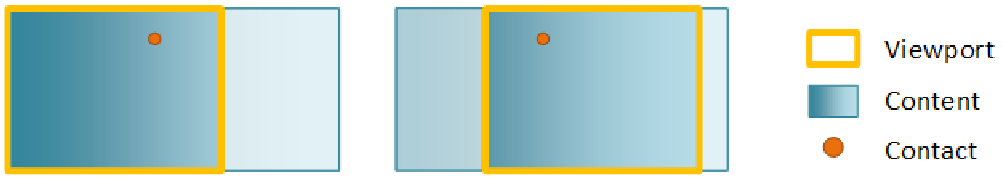
Implementation flexibility

The interfaces included with Direct Manipulation provide comprehensive support for input handling, interaction recognition, feedback notifications, and UI updates. The interfaces also incorporate system services such as DirectComposition.

Basic concepts

The most basic Direct Manipulation implementation consists of a *viewport*, *content*, and *interactions*. The *viewport* is a region that is able to receive and process input from user interactions. It is also the region of the content that is visible to the end-user. The *content* is the actual object that end-users can see and is what moves or scales in response to a user interaction. The primary user *interactions* (also known as *manipulations*) supported by Direct Manipulation are panning and zooming. These interactions apply a translate or scale transform to the content within the viewport, respectively. Multiple viewports (each with their own content) can be configured in a single window to create a rich UI experience.

This figure shows a basic Direct Manipulation implementation before and after panning.



During initialization of Direct Manipulation a **DCompDirectManipulationCompositor** object is instantiated and is associated with Direct Manipulation. This object is a wrapper around DirectComposition, which is the system compositor. The object is responsible for applying the output transforms and driving visual updates.

A contact represents a touch point identified by the **pointerId** provided in the WM/_POINTERDOWN message. When a **WM_POINTERDOWN** message is received, the application calls **SetContact**. The application notifies Direct Manipulationabout the contacts that should be handled and the viewport(s) that should react to those contacts. Keyboard and mouse input have special **pointerId** values so they can be handled appropriately by Direct Manipulation.

In our basic case above, when **SetContact** is called a few things happen:

- When the user performs a pan, a WM/_POINTERCAPTURECHANGED message is sent to the application to notify that the contact has been consumed by Direct Manipulation.
- When the user moves the moves, the viewport fires update events which are used by the DirectComposition wrapper to drive visual updates to the screen. To a user panning in a viewport, the content will appear to move smoothly under the contact.
- When the user lifts the contact, the user sees the content continue to move as it transitions into an inertia animation, gradually decelerating until it reaches its final resting place.

Processing keyboard and mouse input

Direct Manipulation allows keyboard and mouse messages to be forwarded manually from the application UI thread via the **ProcessInput** API such that they can be handled appropriately by Direct Manipulation.

DirectManipulation and the HWND

Direct Manipulation is associated with a Win32 HWND in order to receive and process pointer input messages for that window. As Direct Manipulation computes output values, it makes asynchronous callbacks to the Direct Manipulation Component Object Model (COM) objects that are implemented in the application. These callbacks inform the application about the transform that was applied to the objects. Direct Manipulation is activated on the specified HWND by calling **Activate**.

Supporting documentation

- Viewports and content
- Using multiple viewports in DirectManipulation
- Processing input with DirectManipulation
- Composition engine
- Direct Manipulation Reference
- BUILD 2013: WCL-022: Make your desktop app great with touch, mouse, and pen
- Process touch input with Direct Manipulation sample

Ink input

2/18/2021 • 2 minutes to read • Edit Online

Learn about the various Desktop inking APIs for Windows apps.

In this section

TOPIC	DESCRIPTION
Ink presenter	The ink presenter APIs enable Microsoft Win32 apps to manage the input, processing, and rendering of ink input (standard and modified) through an **InkPresenter** object inserted into the app's DirectComposition visual tree.
Ink renderer	The Ink renderer APIs enable the rendering of ink strokes onto the designated Direct2D device context of a Universal Windows app, instead of the default **InkCanvas** control.

Related topics

Pen and stylus interactions, Ink Analysis sample, Simple inking sample, Complex inking sample

Input Feedback Configuration

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of Input Feedback Configuration in Windows 8. Input Feedback Configuration provides access to visual feedback settings for user interactions.

The collection of topics provided here describe how to use Input Feedback Configuration to optimize the user experience of your app. For example, default visual feedback settings might not be necessary, or desirable, in a game app because of interference or conflicts with the gameplay itself.

In this section

TOPIC	DESCRIPTION
Input Feedback Configuration Reference	The topics in this section provide the reference specifications for Input Feedback Configuration.

Developer audience

The Input Feedback ConfigurationAPIs are designed for developers who are building UI frameworks that provide a consistent touch-optimized user experience across desktop applications.

Related topics

User Interaction, SystemParametersInfo

Input Source Identification

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of support for Input Source Identification in Windows 8. Input Source Identification enables applications to accurately and reliably detect an input source, identify the input type, and provide a user experience that's most appropriate for the input type.

The collection of topics provided here describe how your app can obtain this information and best use it to optimize the user experience for your customer.

In this section

TOPIC	DESCRIPTION
Input Source Identification Reference	The topics in this section provide the reference specifications for Input Source Identification in Windows 8.

Developer audience

The Input Source Identification APIs are designed for developers who are building UI frameworks that provide a consistent touch-optimized user experience across desktop applications.

Related topics

User Interaction

Interaction Context

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of support for Interaction Context in Windows 8. Interaction Context enables Windows 8 applications and UI frameworks to support multiple, concurrent interactions by providing gesture detection and manipulation processing.

The collection of topics provided here describe how your app can use Interaction Context to optimize the user experience for your customer.

In this section

TOPIC	DESCRIPTION
Interaction Context Reference	The topics in this section provide the reference specifications for Interaction Context.

Developer audience

The Interaction Context APIs are designed for developers who are building UI frameworks that provide a consistent touch-optimized user experience across desktop applications.

Related topics

User Interaction

Pointer Device Input Stack

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of the Windows pointer device input stack. The Pointer Device Input Stack APIs provide information about the input devices connected to the system.

In this section

TOPIC	DESCRIPTION
Pointer Device Input Stack Reference	The topics in this section provide the reference specifications for the Pointer Device Input Stack in Windows 8.

Developer audience

The Pointer Device Input StackAPIs are designed for developers who require information about the pointer device being used to interact with their app.

Related topics

User Interaction

Pointer Input Messages and Notifications

2/22/2020 • 2 minutes to read • Edit Online

This section describes input messages, notifications, and associated functionality in Windows applications.

In this section

TOPIC	DESCRIPTION
Pointer Input Message Reference	The topics in this section provide the reference specifications for Pointer Input Messages and Notifications.
Pointer Input Messages and Notifications Glossary	Terms and definitions specific to Pointer Input Messages and Notifications.

Radial controller input

2/18/2021 • 2 minutes to read • Edit Online

Learn about the radial controller APIs for Windows Wheel accessories such as the Surface Dial.

In this section

TOPIC	DESCRIPTION
Radial controller interfaces	The topics contained in this section provide the reference specifications for Radial controller input interfaces.

Related topics

Windows.UI.Input, Surface Dial interactions, Universal Windows Platform samples (C# and C++), Windows classic desktop sample

Text Services Framework (Text Services Framework)

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Microsoft Windows Text Services Framework (TSF) is a system service available as a redistributable for Windows. TSF provides a simple and scalable framework for the delivery of advanced text input and natural language technologies. TSF can be enabled in applications, or as a TSF text service. A TSF text service provides multilingual support and delivers text services such as keyboard processors, handwriting recognition, and speech recognition.

Where applicable

Text Services Framework is applicable for Windows-based computers using text services and Windows XP or later versions of the operating system.

Developer audience

Text Services Framework is designed for use by Component Object Model (COM) programmers using the C/C++ programming languages. Programmers should be familiar with text services for Windows-based computers. Knowledge of handwriting recognition, speech recognition, and programming for multilingual support is recommended.

Run-time requirements

For the latest redistributable, download the Windows 10 platform SDK.

For more information about the requirements of specific API elements, see the Requirements section in the](/windows/win32/tsf/text-services-framework-reference)[TFS reference documentation.

In this section

TOPIC	DESCRIPTION
About Text Services Framework	General information about Text Services Framework.
Using Text Services Framework	Information about using Text Services Framework.
Text Services Framework Reference	Documentation about Text Services Framework interfaces, methods, structures, and other code elements.
Glossary	Alphabetical listing of technical terms used in this documentation.

Additional resources

What You Should Know Before Reading This Guide

For the purposes of the Text Services Framework Help, the term "application" refers to a TSF-enabled application, the term "text service" refers to a TSF text service, and the term "manager" refers to the TSF manager. Each term applies as stated herein unless otherwise specified. Text service providers should provide digital signatures with their binary executables.

Touch Hit Testing

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of support for Touch Hit Testing in Windows. Touch Hit Testing lets you identify an input target based on whether the content area of a UI element falls within the contact area or includes the contact point.

Touch hit testing uses complex contact geometry for the entire touch area rather than a single interpolated x-y coordinate. Using the entire contact geometry greatly improves precision and accuracy when you're identifying the most likely target of touch input.

In this section

TOPIC	DESCRIPTION
Touch Hit Testing Reference	The topics in this section provide the reference specifications for Touch Hit Testing.

Developer audience

The Touch Hit Testing APIs are designed for developers who are building UI frameworks that provide a consistent touch-optimized user experience across desktop applications.

Related topics

User Interaction

Touch Injection

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The topics in this section provide an overview of Touch Injection for Windows applications. The Touch Injection APIs enable Windows developers to programmatically simulate touch input.

In this section

TOPIC	DESCRIPTION
Touch Injection Reference	The topics in this section provide the reference specifications for Touch Injection.

Developer audience

The Touch Injection APIs are designed for developers who are building touch input automation and test frameworks using C++.

Related topics

User Interaction

Legacy User Interaction Features

2/18/2021 • 2 minutes to read • Edit Online

User interaction features for Windows 7 and earlier.

In this section

TOPIC	DESCRIPTION
Touch Input	Describes how to use the Windows Touch API to build touch-enabled Windows-based applications.
Keyboard and Mouse Input	Describes how to capture user input from the keyboard and the mouse.
Multipoint Mouse	Describes the framework that enables you to build applications that support multiple mouse devices concurrently on one computer.
Tablet PC	Describes how to build applications for the Tablet PC, which is a fully functional personal computer geared for pen-enabled, handwriting-enabled, and speech-enabled applications.
DirectInput	Describes how to use the DirectInput API to process data from a joystick, or other game controller. DirectInput is legacy and not available for Windows Store apps. Use XInput instead.
Text Services Framework	Describes how to use the Microsoft Windows Text Services Framework (TSF), which is an API that enables text services to supply text to applications in a way that does not require any detailed knowledge of the applications that produce or receive the text. For example, a text service can provide text input from speech or handwriting input devices.

Touch Input

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Windows supports touch input through the interaction model exposed by the end-to-end platform and user interface framework of Windows Touch.

Developer audience

The features of Windows Touch are designed for use by experienced C/C++ developers who are familiar with Component Object Model (COM), UI programming concepts, and general touch interaction concepts.

Run-time requirements

Windows 7

In this section

TOPIC	DESCRIPTION
About Windows Touch	This topic gives a brief overview of Windows Touch.
Programming Guide	This section covers specific topics about software development with the Windows Touch API targeted to the Windows 7 operating system.
Windows Touch Samples	Describes the path to the Windows Touch samples and gives brief overviews of the samples..
Windows Touch Programming Reference	The following sections describe the functionality of the Windows Touch API.
Windows Touch Glossary	This topic provides the definitions of terms that are used by Windows Touch.

Keyboard and Mouse Input

2/22/2020 • 2 minutes to read • Edit Online

The following sections describe methods of capturing user input.

In This Section

NAME	DESCRIPTION
Keyboard Input	Discusses how the system generates keyboard input and how an application receives and processes that input.
Mouse Input	Discusses how the system provides mouse input to your application and how the application receives and processes that input.
Raw Input	Discusses how the system provides raw input to your application and how an application receives and processes that input.

Tablet PC

2/18/2021 • 2 minutes to read • Edit Online

The Tablet PC is a fully functional personal computer geared for pen-enabled, handwriting-enabled, and speech-enabled applications. The combination of software and hardware in a Tablet PC enables these methods of user interaction and allows for a rich, interactive, and productive computing experience for users.

Tablet PC technology in Windows enables input and output of handwriting and speech data on a Tablet PC as well as interchange of this data with other computers.

Windows Touch, introduced in Windows 7, enables multitouch gestures and an improved touch experience for users.

In This Section

This section contains programming documentation, sample program descriptions, and other resources for software development geared for deployment on a Tablet PC.

- Getting Started

  Shows how to get started with the Tablet PC technology and Microsoft Visual Studio.

- Programming the Tablet PC

  Provides an overview of the programming concepts unique to Tablet PC technology.

- Tablet PC API Reference

  Describes the managed library and Automation API and ink controls, such as the InkEdit and InkPicture controls, that you can use to build applications for Tablet PC and computers that support Windows Touch.

- Appendices

  Contains sections that describe special purpose APIs and references related to Tablet PC.

Related topics

Frequently Asked Questions

Windows and Messages

2/18/2021 • 2 minutes to read • Edit Online

The following sections describe the elements of an application with a Windows-based graphical user interface.

In This Section

NAME	DESCRIPTION
Windows	Discusses windows in general.
Window Classes	Describes the types of window classes, how the system locates them, and the elements that define the default behavior of windows that belong to them.
Window Procedures	Discusses window procedures. Every window has an associated window procedure that processes all messages sent or posted to all windows of the class.
Messages and Message Queues	Describes messages and message queues and how to use them in your applications.
Timers	Discusses timers. A timer is an internal routine that repeatedly measures a specified interval, in milliseconds.
Window Properties	Discusses window properties. A window property is any data assigned to a window.
Configuration	Describes the functions that can be used to control the configuration of system metrics and various system attributes such as double-click time, screen saver time-out, window border width, and desktop pattern.
Hooks	Discusses hooks. A hook is a point in the system message-handling mechanism where an application can install a subroutine to monitor the message traffic.
Multiple Document Interface	Discusses the Multiple Document Interface which is a specification that defines a user interface for applications that enable the user to work with more than one document at the same time.

Desktop Window Manager

2/22/2020 • 2 minutes to read • Edit Online

The desktop composition feature, introduced in Windows Vista, fundamentally changed the way applications display pixels on the screen. When desktop composition is enabled, individual windows no longer draw directly to the screen or primary display device as they did in previous versions of Windows. Instead, their drawing is redirected to off-screen surfaces in video memory, which are then rendered into a desktop image and presented on the display.

Desktop composition is performed by the Desktop Window Manager (DWM). Through desktop composition, DWM enables visual effects on the desktop as well as various features such as glass window frames, 3-D window transition animations, Windows Flip and Windows Flip3D, and high resolution support.

The Desktop Window Manager runs as a Windows service. It can be enabled and disabled through the Administrative Tools Control Panel item, under Services, as Desktop Window Manager Session Manager.

Many of the DWM features can be controlled or accessed by an application through the DWM APIs. The following documentation describes the features and requirements of the DWM APIs.

- DWM Overviews
- DWM Sample Code
- DWM Reference

Dialog Boxes (Dialog Boxes)

2/18/2021 • 6 minutes to read • Edit Online

A dialog box is a temporary window an application creates to retrieve user input. An application typically uses dialog boxes to prompt the user for additional information for menu items. A dialog box usually contains one or more controls (child windows) with which the user enters text, chooses options, or directs the action.

Windows also provides predefined dialog boxes that support common menu items such as **Open** and **Print**. Applications that use these menu items should use the common dialog boxes to prompt for this user input, regardless of the type of application.

In This Section

NAME	DESCRIPTION
About Dialog Boxes	Discusses using dialog boxes in the user interface for your applications.
Dialog Box Programming Considerations	This overview discusses some programming considerations concerning dialog boxes.
Using Dialog Boxes	You use dialog boxes to display information and prompt for input from the user.
Dialog Box Reference	The API Reference
Common Dialog Box Library	Discusses using the common dialog boxes in the user interface for your applications.

Dialog Box Functions

NAME	DESCRIPTION
CreateDialog	Creates a modeless dialog box from a dialog box template resource.
CreateDialogIndirect	Creates a modeless dialog box from a dialog box template in memory.
CreateDialogIndirectParam	Creates a modeless dialog box from a dialog box template in memory. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the *lParam* parameter of the **WM_INITDIALOG** message. An application can use this value to initialize dialog box controls.
CreateDialogParam	Creates a modeless dialog box from a dialog box template resource. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the *lParam* parameter of the **WM_INITDIALOG** message. An application can use this value to initialize dialog box controls.

NAME	DESCRIPTION
DefDlgProc	Calls the default dialog box window procedure to provide default processing for any window messages that a dialog box with a private window class does not process.
DialogBox	Creates a modal dialog box from a dialog box template resource. DialogBox does not return control until the specified callback function terminates the modal dialog box by calling the EndDialog function.
DialogBoxIndirect	Creates a modal dialog box from a dialog box template in memory. DialogBoxIndirect does not return control until the specified callback function terminates the modal dialog box by calling the EndDialog function.
DialogBoxIndirectParam	Creates a modal dialog box from a dialog box template in memory. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the *lParam* parameter of the WM_INITDIALOG message. An application can use this value to initialize dialog box controls.
DialogBoxParam	Creates a modal dialog box from a dialog box template resource. Before displaying the dialog box, the function passes an application-defined value to the dialog box procedure as the *lParam* parameter of the WM_INITDIALOG message. An application can use this value to initialize dialog box controls.
DialogProc	An application-defined callback function used with the CreateDialog and DialogBox families of functions. It processes messages sent to a modal or modeless dialog box. The **DLGPROC** type defines a pointer to this callback function. *DialogProc* is a placeholder for the application-defined function name.
EndDialog	Destroys a modal dialog box, causing the system to end any processing for the dialog box.
GetDialogBaseUnits	Retrieves the system's dialog base units, which are the average width and height of characters in the system font. For dialog boxes that use the system font, you can use these values to convert between dialog template units, as specified in dialog box templates, and pixels. For dialog boxes that do not use the system font, the conversion from dialog template units to pixels depends on the font used by the dialog box.
GetDlgCtrlID	Retrieves the identifier of the specified control.
GetDlgItem	Retrieves a handle to a control in the specified dialog box.
GetDlgItemInt	Translates the text of a specified control in a dialog box into an integer value.
GetDlgItemText	Retrieves the title or text associated with a control in a dialog box.

NAME	DESCRIPTION
GetNextDlgGroupItem	Retrieves a handle to the first control in a group of controls that precedes (or follows) the specified control in a dialog box.
GetNextDlgTabItem	Retrieves a handle to the first control that has the **WS_TABSTOP** style that precedes (or follows) the specified control.
IsDialogMessage	Determines whether a message is intended for the specified dialog box and, if it is, processes the message.
MapDialogRect	Converts the specified dialog box units to screen units (pixels). The function replaces the coordinates in the specified RECT structure with the converted coordinates, which allows the structure to be used to create a dialog box or position a control within a dialog box.
MessageBox	Displays a modal dialog box that contains a system icon, a set of buttons, and a brief application-specific message, such as status or error information. The message box returns an integer value that indicates which button the user clicked.
MessageBoxEx	Creates, displays, and operates a message box. The message box contains an application-defined message and title, plus any combination of predefined icons and push buttons. The buttons are in the language of the system user interface.
MessageBoxIndirect	Creates, displays, and operates a message box. The message box contains application-defined message text and title, any icon, and any combination of predefined push buttons.
SendDlgItemMessage	Sends a message to the specified control in a dialog box.
SetDlgItemInt	Sets the text of a control in a dialog box to the string representation of a specified integer value.
SetDlgItemText	Sets the title or text of a control in a dialog box.

Dialog Box Messages

NAME	DESCRIPTION
DM_GETDEFID	Retrieves the identifier of the default push button control for a dialog box.
DM_REPOSITION	Repositions a top-level dialog box so that it fits within the desktop area. An application can send this message to a dialog box after resizing it to ensure that the entire dialog box remains visible.
DM_SETDEFID	Changes the identifier of the default push button for a dialog box.

Dialog Box Notifications

NAME	DESCRIPTION
WM_CTLCOLORDLG	Sent to a dialog box before the system draws the dialog box. By responding to this message, the dialog box can set its text and background colors using the specified display device context handle.
WM_ENTERIDLE	Sent to the owner window of a modal dialog box or menu that is entering an idle state. A modal dialog box or menu enters an idle state when no messages are waiting in its queue after it has processed one or more previous messages.
WM_GETDLGCODE	Sent to the window procedure associated with a control. By default, the system handles all keyboard input to the control; the system interprets certain types of keyboard input as dialog box navigation keys. To override this default behavior, the control can respond to the WM_GETDLGCODE message to indicate the types of input it wants to process itself.
WM_INITDIALOG	Sent to the dialog box procedure immediately before a dialog box is displayed. Dialog box procedures typically use this message to initialize controls and carry out any other initialization tasks that affect the appearance of the dialog box.
WM_NEXTDLGCTL	Sent to a dialog box procedure to set the keyboard focus to a different control in the dialog box.

Dialog Box Structures

NAME	DESCRIPTION
DLGITEMTEMPLATE	Defines the dimensions and style of a control in a dialog box. One or more of these structures are combined with a DLGTEMPLATE structure to form a standard template for a dialog box.
DLGITEMTEMPLATEEX	Describes an extended dialog box. For a description of the format of an extended dialog box template, see DLGTEMPLATEEX.
DLGTEMPLATE	Defines the dimensions and style of a dialog box. This structure, always the first in a standard template for a dialog box, also specifies the number of controls in the dialog box and therefore specifies the number of subsequent DLGITEMTEMPLATE structures in the template.
DLGTEMPLATEEX	An extended dialog box template begins with a **DLGTEMPLATEEX** header that describes the dialog box and specifies the number of controls in the dialog box. For each control in a dialog box, an extended dialog box template has a block of data that uses the DLGITEMTEMPLATEEX format to describe the control.
MSGBOXPARAMS	Contains information used to display a message box. The MessageBoxIndirect function uses this structure.

Menus and Other Resources

11/2/2020 • 2 minutes to read • Edit Online

A *resource* is binary data that you can add to the executable file of a Windows-based application. A resource can be either standard or defined. The data in a *standard resource* describes an icon, cursor, menu, dialog box, bitmap, enhanced metafile, font, accelerator table, message-table entry, string-table entry, or version information. An *application-defined resource*, also called a *custom resource*, contains any data required by a specific application.

In This Section

NAME	DESCRIPTION
Introduction to Resources	Provides an overview of resources.
Carets	Discusses carets, which are blinking lines, blocks, or bitmaps in the client area of a window.
Cursors	Discusses cursors, which are small pictures whose location on the screen is controlled by a pointing device, such as a mouse, pen, or trackball.
Icons	Discusses icons, which are bitmap images combined with a mask to create transparent areas in the picture.
Keyboard Accelerators	Discusses keyboard accelerators, which are keystrokes that provide access to the commands for an application.
Menus	Discusses menus.
Strings	Discusses the string functions.
Version Information	Discusses the version-information resource.
Resource Compiler	Discusses the resource compiler, Rc.exe, and resource-definition files.
Package resource indexing (PRI) reference	A set of APIs for working with a resource indexer. A resource indexer is used to generate package resource index (PRI) files for a UWP app.

For more information about creating message resources, see Message Compiler.

Data Exchange

2/22/2020 • 2 minutes to read • Edit Online

The following sections describe basic methods of exchanging data.

Data Exchange

NAME	DESCRIPTION
Clipboard	Discusses the clipboard. The clipboard is a set of functions and messages that enable applications to transfer data.
Data Copy	Discusses how to use the data copy message to transfer data from one application to another.
Dynamic Data Exchange	Discusses implementing dynamic data exchange for applications that cannot use the Dynamic Data Exchange Management Library (DDEML).
Dynamic Data Exchange Management Library	Discusses dynamic data exchange, which is a form of interprocess communication that uses shared memory to exchange data between applications.

High DPI Desktop Application Development on Windows

11/2/2020 • 15 minutes to read • Edit Online

This content is targeted at developers who are looking to update desktop applications to handle display scale factor (dots per inch, or DPI) changes dynamically, allowing their applications to be crisp on any display they're rendered on.

To start, if you're creating a new Windows app from scratch, it is highly recommended that you create a Universal Windows Platform (UWP) application. UWP applications automatically—and dynamically—scale for each display that they're running on.

Desktop applications using older Windows programming technologies (raw Win32 programming, Windows Forms, Windows Presentation Framework (WPF), etc.) are unable to automatically handle DPI scaling without additional developer work. Without such work, applications will appear blurry or incorrectly-sized in many common usage scenarios. This document provides context and information about what is involved in updating a desktop application to render correctly.

Display Scale Factor & DPI

As display technology has progressed, display panel manufacturers have packed an increasing number of pixels into each unit of physical space on their panels. This has resulted in the dots per inch (DPI) of modern display panels being much higher than they have historically been. In the past, most displays had 96 pixels per linear inch of physical space (96 DPI); in 2017, displays with nearly 300 DPI or higher are readily available.

Most legacy desktop UI frameworks have built-in assumptions that the display DPI will not change during the lifetime of the process. This assumption no longer holds true, with display DPIs commonly changing several times throughout an application process's lifetime. Some common scenarios where the display scale factor/DPI changes are:

- Multiple-monitor setups where each display has a different scale factor and the application is moved from one display to another (such as a 4K and a 1080p display)
- Docking and undocking a high DPI laptop with a low-DPI external display (or vice versa)
- Connecting via Remote Desktop from a high DPI laptop/tablet to a low-DPI device (or vice versa)
- Making a display-scale-factor settings change while applications are running

In these scenarios, UWP applications redraw themselves for the new DPI automatically. By default, and without additional developer work, desktop applications do not. Desktop applications that don't do this extra work to respond to DPI changes may appear blurry or incorrectly-sized to the user.

DPI Awareness Mode

Desktop applications must tell Windows if they support DPI scaling. By default, the system considers desktop applications DPI unaware and bitmap-stretches their windows. By setting one of the following available DPI awareness modes, applications can explicitly tell Windows how they wish to handle DPI scaling:

DPI Unaware

DPI unaware applications render at a fixed DPI value of 96 (100%). Whenever these applications are run on a screen with a display scale greater than 96 DPI, Windows will stretch the application bitmap to the expected physical size. This results in the application appearing blurry.

System DPI Awareness

Desktop applications that are system DPI aware typically receive the DPI of the primary connected monitor as of the time of user sign-in. During initialization, they lay out their UI appropriately (sizing controls, choosing font sizes, loading assets, etc.) using that System DPI value. As such, System DPI-aware applications are not DPI scaled (bitmap stretched) by Windows on displays rendering at that single DPI. When the application is moved to a display with a different scale factor, or if the display scale factor otherwise changes, Windows will bitmap scale the application's windows, making them appear blurry. Effectively, System DPI-aware desktop applications only render crisply at a single display scale factor, becoming blurry whenever the DPI changes.

Per-Monitor and Per-Monitor (V2) DPI Awareness

It is recommended that desktop applications be updated to use per-monitor DPI awareness mode, allowing them to immediately render correctly whenever the DPI changes. When an application reports to Windows that it wants to run in this mode, Windows will not bitmap stretch the application when the DPI changes, instead sending WM_DPICHANGED to the application window. It is then the complete responsibility of the application to handle resizing itself for the new DPI. Most UI frameworks used by desktop applications (Windows common controls (comctl32), Windows Forms, Windows Presentation Framework, etc.) do not support automatic DPI scaling, requiring developers to resize and reposition the contents of their windows themselves.

There are two versions of Per-Monitor awareness that an application can register itself as: version 1 and version 2 (PMv2). Registering a process as running in PMv2 awareness mode results in:

1. The application being notified when the DPI changes (both the top-level and child HWNDs)
2. The application seeing the raw pixels of each display
3. The application never being bitmap scaled by Windows
4. Automatic non-client area (window caption, scroll bars, etc.) DPI scaling by Windows
5. Win32 dialogs (from CreateDialog) automatically DPI scaled by Windows
6. Theme-drawn bitmap assets in common controls (checkboxes, button backgrounds, etc.) being automatically rendered at the appropriate DPI scale factor

When running in Per-Monitor v2 Awareness mode, applications are notified when their DPI has changed. If an application does not resize itself for the new DPI, the application UI will appear too small or too large (depending on the difference in the previous and new DPI values).

> **NOTE**
>
> Per-Monitor V1 (PMv1) awareness is very limited. It is recommended that applications use PMv2.

The following table shows how applications will render under different scenarios:

DPI AWARENESS MODE	WINDOWS VERSION INTRODUCED	APPLICATION'S VIEW OF DPI	BEHAVIOR ON DPI CHANGE
Unaware	N/A	All displays are 96 DPI	Bitmap-stretching (blurry)
System	Vista	All displays have the same DPI (the DPI of the primary display at the time the current user session was started)	Bitmap-stretching (blurry)

DPI AWARENESS MODE	WINDOWS VERSION INTRODUCED	APPLICATION'S VIEW OF DPI	BEHAVIOR ON DPI CHANGE
Per-Monitor	8.1	The DPI of the display that the application window is primarily located on	• Top-level HWND is notified of DPI change • No DPI scaling of any UI elements.
Per-Monitor V2	Windows 10 Creators Update (1703)	The DPI of the display that the application window is primarily located on	• Top-level and child HWNDs are notified of DPI change Automatic DPI scaling of: • Non-client area • Theme-drawn bitmaps in common controls (comctl32 V6) • Dialogs (CreateDialog)

Per Monitor (V1) DPI Awareness

Per-Monitor V1 DPI awareness mode (PMv1) was introduced with Windows 8.1. This DPI awareness mode is very limited and only offers the functionality listed below. It is recommended that desktop applications use Per-Monitor v2 awareness mode, supported on Windows 10 1703 or above.

The initial support for per-monitor awareness only offered applications the following:

1. Top-level HWNDs are notified of a DPI change and provided a new suggested size
2. Windows will not bitmap stretch the application UI
3. The application sees all displays in physical pixels (see virtualization)

On Windows 10 1607 or above, PMv1 applications may also call EnableNonClientDpiScaling during WM_NCCREATE to request that Windows correctly scale the window's non-client area.

Per Monitor DPI Scaling Support by UI Framework / Technology

The table below shows the level of per-monitor DPI awareness support offered by various Windows UI frameworks as of Windows 10 1703:

FRAMEWORK / TECHNOLOGY	SUPPORT	OS VERSION	DPI SCALING HANDLED BY	FURTHER READING
Universal Windows Platform (UWP)	Full	1607	UI framework	Universal Windows Platform (UWP)

FRAMEWORK / TECHNOLOGY	SUPPORT	OS VERSION	DPI SCALING HANDLED BY	FURTHER READING
Raw Win32/Common Controls V6 (comctl32.dll)	• DPI change notification messages sent to all HWNDs • Theme-drawn assets render correctly in common controls • Automatic DPI scaling for dialogs	1703	Application	GitHub Sample
Windows Forms	Limited automatic per-monitor DPI scaling for some controls	1703	UI framework	High DPI Support in Windows Forms
Windows Presentation Framework (WPF)	Native WPF applications will DPI scale WPF hosted in other frameworks and other frameworks hosted in WPF do not automatically scale	1607	UI framework	GitHub Sample
GDI	None	N/A	Application	See GDI High-DPI Scaling
GDI+	None	N/A	Application	See GDI High-DPI Scaling
MFC	None	N/A	Application	N/A

Updating Existing Applications

In order to update an existing desktop application to handle DPI scaling properly, it needs to be updated such that, at a minimum, the important parts of its UI are updated to respond to DPI changes.

Most desktop applications run under system DPI awareness mode. System-DPI-aware applications typically scale to the DPI of the primary display (the display that the system tray was located on at the time the Windows session was started). When the DPI changes, Windows will bitmap stretch the UI of these applications, which often results in them being blurry. When updating a System DPI-aware application to become per-monitor-DPI aware, the code which handles UI layout needs to be updated such that it is performed not only during application initialization, but also whenever a DPI change notification (WM_DPICHANGED in the case of Win32) is received. This typically involves revisiting any assumptions in the code that the UI only needs to be scaled once.

Also, in the case of Win32 programming, many Win32 APIs do not have any DPI or display context so they will only return values relative to the System DPI. It can be useful to grep through your code to look for some of these APIs and replace them with DPI-aware variants. Some of the common APIs that have DPI-aware variants are:

SINGLE DPI VERSION	PER-MONITOR VERSION
GetSystemMetrics	GetSystemMetricsForDpi
AdjustWindowRectEx	AdjustWindowRectExForDpi
SystemParametersInfo	SystemParametersInfoForDpi
GetDpiForMonitor	GetDpiForWindow

It is also a good idea to search for hard-coded sizes in your codebase that assume a constant DPI, replacing them with code that correctly accounts for DPI scaling. Below is an example that incorporates all of these suggestions:

Example:

The example below shows a simplified Win32 case of creating a child HWND. The call to CreateWindow assumes that the application is running at 96 DPI, and neither the button's size nor position will be correct at higher DPIs:

```
case WM_CREATE:
{
    // Add a button
    HWND hWndChild = CreateWindow(L"BUTTON", L"Click Me",
        WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
        50,
        50,
        100,
        50,
        hWnd, (HMENU)NULL, NULL, NULL);
}
```

The updated code below shows:

1. The window-creation code DPI scaling the position and size of the child HWND for the DPI of its parent window
2. Responding to DPI change by repositioning and resizing the child HWND
3. Hard-coded sizes removed and replaced with code that responds to DPI changes

```
#define INITIALX_96DPI 50
#define INITIALY_96DPI 50
#define INITIALWIDTH_96DPI 100
#define INITIALHEIGHT_96DPI 50


// DPI scale the position and size of the button control
void UpdateButtonLayoutForDpi(HWND hWnd)
{
    int iDpi = GetDpiForWindow(hWnd);
    int dpiScaledX = MulDiv(INITIALX_96DPI, iDpi, 96);
    int dpiScaledY = MulDiv(INITIALY_96DPI, iDpi, 96);
    int dpiScaledWidth = MulDiv(INITIALWIDTH_96DPI, iDpi, 96);
    int dpiScaledHeight = MulDiv(INITIALHEIGHT_96DPI, iDpi, 96);
    SetWindowPos(hWnd, hWnd, dpiScaledX, dpiScaledY, dpiScaledWidth, dpiScaledHeight, SWP_NOZORDER |
SWP_NOACTIVATE);
}

...

case WM_CREATE:
{
    // Add a button
    HWND hWndChild = CreateWindow(L"BUTTON", L"Click Me",
        WS_CHILD|WS_VISIBLE|BS_PUSHBUTTON,
        0,
        0,
        0,
        0,
        hWnd, (HMENU)NULL, NULL, NULL);
    if (hWndChild != NULL)
    {
        UpdateButtonLayoutForDpi(hWndChild);
    }
}
break;

case WM_DPICHANGED:
{
    // Find the button and resize it
    HWND hWndButton = FindWindowEx(hWnd, NULL, NULL, NULL);
    if (hWndButton != NULL)
    {
        UpdateButtonLayoutForDpi(hWndButton);
    }
}
break;
```

When updating a System DPI-aware application, some common steps to follow are:

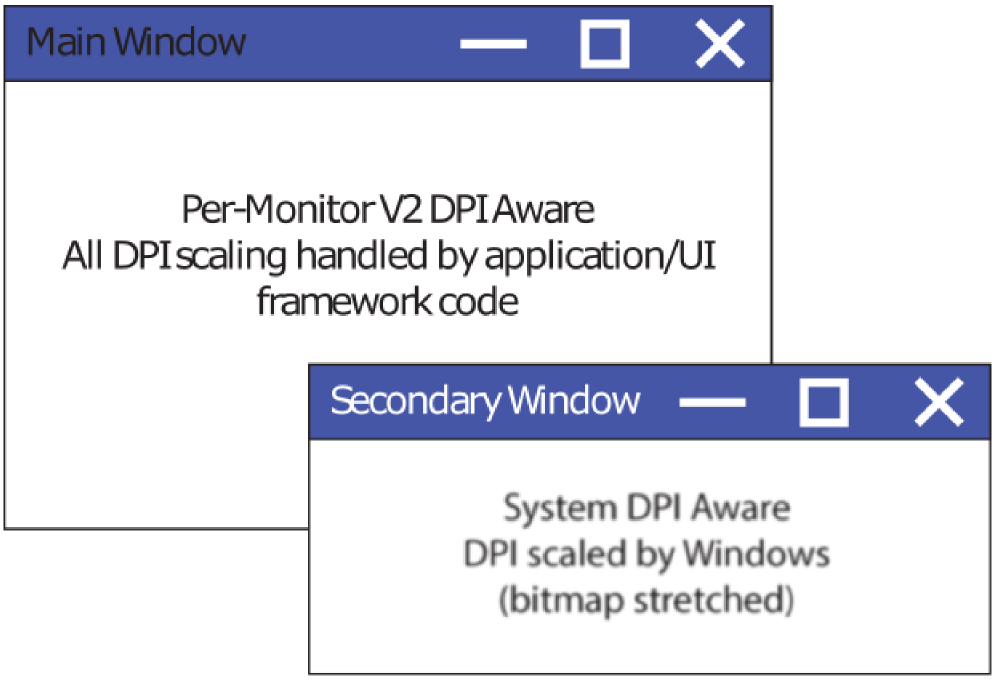
1. Mark the process as per-monitor DPI aware (V2) using an application manifest (or other method, depending on the UI framework(s) used).
2. Make UI layout logic reusable and move it out of application-initialization code such that it can be reused when a DPI change occurs (WM_DPICHANGED in the case of Windows (Win32) programming).
3. Invalidate any code that assumes that DPI-sensitive data (DPI/fonts/sizes/etc.) never need to be updated. It is a very common practice to cache font sizes and DPI values at process initialization. When updating an application to become per-monitor DPI aware, DPI-sensitive data must be reevaluated whenever a new DPI is encountered.
4. When a DPI change occurs, reload (or re-rasterize) any bitmap assets for the new DPI or, optionally, bitmap stretch the currently loaded assets to the correct size.
5. Grep for APIs that are not Per-Monitor DPI aware and replace them with Per-Monitor DPI-aware APIs (where applicable). Example: replace GetSystemMetrics with GetSystemMetricsForDpi.

6. Test your application on a multiple-display/multi-DPI system.
7. For any top-level windows in your application that you are unable to update to properly DPI scale, use mixed-mode DPI scaling (described below) to allow bitmap stretching of these top-level windows by the system.

Mixed-Mode DPI Scaling (Sub-Process DPI Scaling)

When updating an application to support per-monitor DPI awareness, it can sometimes become impractical or impossible to update every window in the application in one go. This can simply be due to the time and effort required to update and test all UI, or because you do not own all of the UI code that you need to run (if your application perhaps loads third-party UI). In these situations, Windows offers a way to ease into the world of per-monitor awareness by letting you run some of your application windows (top-level only) in their original DPI-awareness mode while you focus your time and energy updating the more important parts of your UI.

Below is an illustration of what this could look like: you update your main application UI ("Main Window" in the illustration) to run with per-monitor DPI awareness while you run other windows in their existing mode ("Secondary Window").



Prior to the Windows 10 Anniversary Update (1607), the DPI awareness mode of a process was a process-wide property. Beginning in the Windows 10 Anniversary Update, this property can now be set per **top-level** window. (**Child** windows must continue to match the scaling size of their parent.) A top-level window is defined as a window with no parent. This is typically a "regular" window with minimize, maximize, and close buttons. The scenario that sub-process DPI awareness is intended for is to have secondary UI scaled by Windows (bitmap stretched) while you focus your time and resources on updating your primary UI.

To enable sub-process DPI awareness, call SetThreadDpiAwarenessContext before and after any window creation calls. The window that is created will be associated with the DPI awareness that you set via SetThreadDpiAwarenessContext. Use the second call to restore the current thread s DPI awareness.

While using sub-process DPI scaling enables you to rely on Windows to do some of the DPI scaling for your application, it can increase the complexity of your application. It is important that you understand the drawbacks of this approach and nature of the complexities that it introduces. For more information about sub-process DPI awareness, see Mixed-Mode DPI Scaling and DPI-aware APIs.

Testing Your Changes

After you have updated your application to become per-monitor DPI aware, it is important to validate your application properly responds to DPI changes in a mixed-DPI environment. Some specifics to test include:

1. Moving application windows back and forth between displays of different DPI values
2. Starting your application on displays of different DPI values
3. Changing the scale factor for your monitor while the application is running
4. Changing the display that you use as the primary display, *signing out of Windows*, then re-testing your application after signing back in. This is particularly useful in finding code that uses hard-coded sizes/dimensions.

Common Pitfalls (Win32)

Not using the suggested rectangle that is provided in WM_DPICHANGED

When Windows sends your application window a **WM_DPICHANGED** message, this message includes a suggested rectangle that you should use to resize your window. It is critical that your application use this rectangle to resize itself, as this will:

1. Ensure that the mouse cursor will stay in the same relative position on the Window when dragging between displays
2. Prevent the application window from getting into a recursive dpi-change cycle where one DPI change triggers a subsequent DPI change, which triggers yet another DPI change.

If you have application-specific requirements that prevent you from using the suggested rectangle that Windows provides in the WM_DPICHANGED message, see **WM_GETDPISCALEDSIZE**. This message can be used to give Windows a desired size you'd like used once the DPI change has occurred, while still avoiding the issues described above.

Lack of documentation about virtualization

When an HWND or process is running as either DPI unaware or system DPI aware, it can be bitmap stretched by Windows. When this happens, Windows scales and converts DPI-sensitive information from some APIs to the coordinate space of the calling thread. For example, if a DPI-unaware thread queries the screen size while running on a high-DPI display, Windows will virtualize the answer given to the application as if the screen were in 96 DPI units. Alternatively, when a System DPI-aware thread is interacting with a display at a different DPI than was in use when the current user's session was started, Windows will DPI-scale some API calls into the coordinate space that the HWND would be using if it were running at its original DPI scale factor.

When you update your desktop application to DPI scale properly, it can difficult to know which API calls can return virtualized values based on the thread context; this information is not currently sufficiently documented by Microsoft. Be aware that if you call any system API from a DPI-unaware or system-DPI-aware thread context, the return value might be virtualized. As such, make sure your thread is running in the DPI context you expect when interacting with the screen or individual windows. When temporarily changing a thread's DPI context using SetThreadDpiAwarenessContext, be sure to restore the old context when you're done to avoid causing incorrect behavior elsewhere in your application.

Many Windows APIs do not have an DPI context

Many legacy Windows APIs do not include a DPI or HWND context as part of their interface. As a result, developers often have to do additional work to handle the scaling of any DPI-sensitive information, such as sizes, points, or icons. As an example, developers using LoadIcon must either bitmap stretch loaded icons or use alternate APIs to load correctly-sized icons for the appropriate DPI, such as LoadImage.

Forced reset of process-wide DPI awareness

In general, the DPI awareness mode of your process cannot be changed after process initialization. Windows can, however, forcibly change the DPI awareness mode of your process if you attempt to break the requirement that all HWNDs in a window tree have the same DPI awareness mode. On all versions of Windows, as of Windows 10 1703, it is not possible to have different HWNDs in an HWND tree run in different DPI awareness modes. If you attempt to create a child-parent relationship that breaks this rule, the DPI awareness of the entire process can be reset. This can be triggered by:

1. A CreateWindow call where the passed in parent window is of a different DPI awareness mode than the calling thread.
2. A SetParent call where the two windows are associated with different DPI awareness modes.

The table below shows what happens if you attempt to violate this rule:

OPERATION	WINDOWS 8.1	WINDOWS 10 (1607 AND EARLIER)	WINDOWS 10 (1703 AND LATER)
CreateWindow (In-Proc)	N/A	**Child inherits** (mixed mode)	**Child inherits** (mixed mode)
CreateWindow (Cross-Proc)	**Forced reset** (of caller's process)	**Child inherits** (mixed mode)	**Forced reset** (of caller's process)
SetParent (In-Proc)	N/A	**Forced reset** (of current process)	**Fail** (ERROR_INVALID_STATE)
SetParent (Cross-Proc)	**Forced reset** (of child window's process)	**Forced reset** (of child window's process)	**Forced reset** (of child window's process)

Related topics

High DPI API Reference

Mixed-Mode DPI Scaling and DPI-aware APIs.

Windows Animation Manager

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Windows Animation Manager (Windows Animation) enables rich animation of user interface elements. It is designed to simplify the process of adding animation to an application's user interface and to enable developers to implement animations that are smooth, natural, and interactive.

The animation framework manages the scheduling and execution of animations. It supplies a library of useful mathematical functions for specifying the behavior of a UI element over time, and also enables developers to implement custom functions that provide additional behaviors.

Windows Animation does not perform the rendering; it can be used with any graphics platform, including Direct2D, Direct3D, or GDI+.

In this section

TOPIC	DESCRIPTION
Windows Animation Development Guide	The developer guide provides an overview of Windows Animation and provides sample code that covers the basic animation tasks.
Windows Animation Reference	The topics contained in this section provide the reference specifications for the Windows Animation Manager.
Windows Animation Samples	The topics contained in this section provide details about the code samples that support the Windows Animation Manager documentation.
Windows Animation Glossary	This glossary contains terms and acronyms of interest to developers using the Windows Animation Manager.

Developer audience

Windows Animation is designed for use by experienced C/C++ developers who are familiar with COM, UI programming concepts, and general animation concepts.

Run-time requirements

The Windows Animation Manager was introduced in Windows 7.

Applications that require support for Windows Animation Manager on Windows Vista can utilize the Platform Update for Windows Vista. Applications that require Platform Update for Windows Vista can have Windows Update verify that it is installed, or download and install it in the background otherwise. For more information, see About Platform Update for Windows Vista.

Additional resources

- Windows Scenic Animation Overview (video)

- Inside Windows 7: Animation Manager Deep Dive and Tutorial (video)
- Windows Animation - Advanced Topics (video)
- Windows Animation Manager Sample Code

Related topics

Animation Overview (.NET Framework)

Windows Controls

2/22/2020 • 2 minutes to read • Edit Online

Purpose

A control is a child window that an application uses in conjunction with another window to enable user interaction. Controls are most often used within dialog boxes, but they can also be used in other windows. Controls within dialog boxes provide the user with a way to type text, choose options, and initiate actions. Controls in other windows provide a variety of services, such as letting the user choose commands, view status, and view and edit text. This documentation describes the controls provided by Windows and the programming elements used to create and manipulate them.

For a list of all Windows controls, including a link to comprehensive overview and reference information for each control, see Control Library.

Developer audience

Controls are designed for use by C/C++ developers and UI designers. In general, developers need a moderate level of understanding about UI programming concepts, Windows API programming, and Unicode.

Run-time requirements

Support for controls is provided by User32.dll and Comctl32.dll. For more information, see Common Control Versions.

In this section

TOPIC	DESCRIPTION
About Common Controls	Provides general information that is common to all controls that are supported by Comctl32.dll.
Control Messages	Explains how Windows messages are used to communicate with controls.
Custom Controls	Describes various ways of creating custom controls.
Subclassing Controls	Describes a way to customize a control by changing its features or adding new ones.
Custom Draw	Describes a service, provided by some controls, that applications can use to customize various aspects of the control's appearance.
Security Considerations: Microsoft Windows Controls	Provides information about security considerations related to the Windows controls.
Control Library	Provides overviews and reference information about each control supported by User32.dll and Comctl32.dll.

TOPIC	DESCRIPTION
General Control Reference	Provides reference information about programming elements that apply to multiple controls, not just to a specific control.
Control Spy v2.0	Describes Control Spy, a tool that helps developers understand common controls.
Visual Styles	Describes how the appearance of controls can change depending on the visual style chosen by the user.
Theme File Format	Discusses the format of Theme (.theme) files used in Windows 7 and Windows Vista.

Windows Ribbon Framework

2/18/2021 • 2 minutes to read • Edit Online

The Windows Ribbon framework is a rich command presentation system that provides a modern alternative to the layered menus, toolbars, and task panes of traditional Windows applications. Similar in functionality and appearance to the Microsoft Office 2007 Fluent user interface, the Ribbon framework is composed of a ribbon command bar that exposes the major features of an application through a series of tabs at the top of an application window, and a context menu system.

In this section

TOPIC	DESCRIPTION
Ribbon Framework Overviews	The topics contained in this section explore the fundamentals of the Ribbon framework.
Ribbon Framework Developer Guides	The topics contained in this section describe specific aspects of the Windows Ribbon framework.
Ribbon Framework Control Library	The topics contained in this section describe the set of controls that are included with the Ribbon framework. The controls listed here are the UI objects in a ribbon that expose Command functionality.
Ribbon Framework Reference	The topics contained in this section provide the Reference specifications for the Ribbon framework.
Ribbon Framework Samples	The topics contained in this section provide details about the code samples that support the Windows Ribbon framework documentation.
Ribbon Framework Glossary	

Developer Audience

The Windows Ribbon framework is designed for use by C/C++ developers and UI designers.

Recommended proficiencies:

- COM programming
- Windows API programming
- XML/XAML programming

Recommended foundational knowledge:

- UI programming concepts
- General UI concepts

Minimum Requirements

REQUIREMENT	VALUE
Minimum supported client	Windows 7 Windows Vista with Service Pack 2 (SP2) and Platform Update for Windows Vista
Minimum supported server	Windows Server 2008 R2 Windows Server 2008 with SP2 and Platform Update for Windows Server 2008
Windows Software Development Kit (SDK)	7.0
Header and IDL files	uiribbon.h, uiribbon.idl

NOTE

The Platform Update for Windows Vista and Platform Update for Windows Server 2008 are sets of run-time libraries that enable developers to target Windows Ribbon applications to both Windows Vista and Windows Server 2008. The platform updates will be available to all Windows Vista and Windows Server 2008 customers through Windows Update. Third-party applications that require Platform Update for Windows Vista or Platform Update for Windows Server 2008 can have Windows Update detect whether the required updated is installed; if it is not, Windows Update will download and install it in the background.

See Also

Component Object Model (COM)

Windows API

XAML

Desktop Environment

2/18/2021 • 2 minutes to read • Edit Online

This section provides guidance for integrating and extending the desktop user-facing features of Windows. You can learn how to ensure your apps and file formats appear and behave properly in Start, in the Taskbar, on the desktop, and in File Explorer. You can also ensure your file formats and data stores are indexable and searchable.

In this section

TOPIC	DESCRIPTION
Windows Shell	The Windows desktop UI provides users with access to a wide variety of objects necessary for running applications and managing the operating system. The most numerous and familiar of these objects are the folders and files that reside on computer disk drives. There are also a number of virtual objects that allow the user to perform tasks such as sending files to remote printers or accessing the Recycle Bin. The Shell organizes these objects into a hierarchical namespace and provides users and applications with a consistent and efficient way to access and manage objects.
Windows Property System	The Windows Property System is an extensible read/write system of data definitions that provides a uniform way of expressing metadata about Shell items. The Windows Property system enables you to store and retrieve metadata for Shell items. A Shell item is any single piece of content, such as a file, folder, email, or contact. A property is an individual piece of metadata associated with a Shell item.
Windows Search	Windows Search is a desktop search platform that has instant search capabilities for most common file and data types such as email, contacts, calendar appointments, documents, photos, multimedia, and other formats that can be extended by third party developers. These capabilities enable users to find, manage, and organize the increasing amount of data common in home and enterprise environments.
Window Stations and Desktops	Windows provides three main categories of objects: user interface, graphics device interface (GDI), and kernel. Kernel objects are securable, while user objects and GDI objects are not. Therefore, to provide additional security, user interface objects are managed using window stations and desktops, which themselves are securable objects.
Windows Help	Describes the Help technologies available in Windows.
Consoles	Consoles manage input and output (I/O) for character-mode applications (applications that do not provide their own graphical user interface).

Related topics

Windows Application UI Development

Windows Shell

2/18/2021 • 2 minutes to read • Edit Online

The Windows UI provides users with access to a wide variety of objects necessary for running applications and managing the operating system. The most numerous and familiar of these objects are the folders and files that reside on computer disk drives. There are also a number of virtual objects that allow the user to perform tasks such as sending files to remote printers or accessing the Recycle Bin. The Shell organizes these objects into a hierarchical namespace and provides users and applications with a consistent and efficient way to access and manage objects.

Shell Development Scenarios

The following development scenarios relate to application development:

- Extending the Shell, which consists of creating a data source (versus consuming the Shell data model)
- Implementing a subset of the Shell data source tasks
- Supporting libraries and item views in Windows Explorer
- Using the common file dialog
- Implementing Control Panel items
- Managing notifications

The following development scenarios relate to file format ownership:

- Implementing a subset of the Shell data source tasks
- Implementing any handler
- Supporting desktop search

The following development scenarios relate to data storage ownership:

- Supporting desktop search and OpenSearch
- Implementing a subset of the Shell data source tasks (virtual folders)
- Supporting libraries in Windows Explorer

The following development scenario relates to device support:

- Auto run and auto play

Windows Shell SDK Documentation

This documentation is broken into three major sections:

- The Shell Developer's Guide provides conceptual material about how the Shell works and how to use the Shell's API in your application.
- The Shell Reference section documents programming elements that make up the various Shell APIs.
- Shell Samples provides links to related code samples.

The following table provides an outline of the Shell Reference section. Unless otherwise noted, all programming elements are documented in unmanaged C++.

SECTION	DESCRIPTION
Shell Classes	This section describes select Windows Shell classes.
Shell Interfaces	This section describes the Windows Shell Component Object Model (COM) interfaces.
Shell Functions	This section describes the Windows Shell functions.
Shell Callback Functions	This section describes the Windows Shell callback functions templates.
Shell Constants, Enumerations, and Flags	This section describes the Windows Shell constants, enumerations, and flags used in the Shell APIs.
Shell Lightweight Utility Functions	This section describes the Windows Shell lightweight utility functions provided in Shlwapi.dll.
Shell Macros	This section describes the Windows Shell utility macros.
Shell Messages and Notifications	This section describes the messages and notifications sent by elements of the Windows Shell.
Shell Objects for Scripting and Microsoft Visual Basic	This section describes the Windows objects implemented by the Shell for use in scripting and Microsoft Visual Basic.
Shell Objects for C++	This section describes the C++ Windows objects implemented by the Shell.
Shell Schemas	This section describes library, property, and transfer manifest schemas used by the Windows Shell.
Shell Structures	This section describes the Windows Shell structures used in the Shell APIs.

Windows Property System

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Windows Property System is an extensible read/write system of data definitions that provides a uniform way of expressing metadata about Shell items. The Windows Property system in Windows Vista and later enables you to store and retrieve metadata for Shell items. A Shell item is any single piece of content, such as a file, folder, email, or contact. A property is an individual piece of metadata associated with a Shell item.

Developer audience

Before you start reading the Windows Property System SDK documentation, you should have a fundamental understanding of the following:

- Component Object Model (COM)
- Shell Namespace programming

For an introduction to COM, see COM Fundamentals. For an introduction to Shell Namespace programming, see Getting Started with the Shell Namespace.

For uses of the Windows Property System, see Property System Overview: Development Scenarios.

Run-time requirements

The supported run-time environment for using the Windows Property System is Windows Vista or later and the Windows Software Development Kit (SDK). Windows XP and Microsoft Windows Desktop Search (WDS) 3.0 or later also include a subset of the Windows Property System. For the Windows 7 or updated Windows Vista SDK download, see the Windows SDK.

In this section

- Property System Overview
- Windows Property System Developer's Guide
- Property System Reference
- Property System Code Samples

Windows Search

2/18/2021 • 2 minutes to read • Edit Online

Windows Search is a desktop search platform that has instant search capabilities for most common file and data types such as email, contacts, calendar appointments, documents, photos, multimedia, and other formats that can be extended by third party developers. These capabilities enable users to find, manage, and organize the increasing amount of data common in home and enterprise environments.

In this section

- Windows Search Overview
- Windows Search Developer's Guide
- Windows Search Reference
- Windows Search Code Samples
- Related Search Technologies
- Federated Search in Windows
- Windows Search Glossary

Additional Resources

- For community-supported question and discussion message boards on Search technologies, see MSDN Forum: Windows Desktop Search Development.
- To download the Search Code Samples:
  - Windows Search Samples
- To download the Windows SDK:
  - For Windows 10: Windows 10 SDK
  - For Windows 7: Windows SDK for Windows 7 and .NET Framework

Window Stations and Desktops

11/2/2020 • 2 minutes to read • Edit Online

Windows provides three main categories of objects: user interface, graphics device interface (GDI), and kernel. Kernel objects are securable, while user objects and GDI objects are not. Therefore, to provide additional security, user interface objects are managed using window stations and desktops, which themselves are securable objects.

For more information, see the following topics:

- About Window Stations and Desktops
- Window Station and Desktop Reference

Application Installation and Servicing

2/18/2021 • 2 minutes to read • Edit Online

Make use of available APIs and services provided by Windows to install, manage, and service your desktop apps.

In this section

TOPIC	DESCRIPTION
Games Explorer	Describes how to use Games Explorer to display games on Windows.
Isolated Applications and Side-by-side Assemblies	Isolated Applications and Side-by-side Assemblies is a Microsoft Windows solution that reduces versioning conflicts in Windows-client applications.
Packaging, deployment, and query of Windows apps	Programmatically create Windows app packages (including UWP and desktop apps), and install, update, query, and uninstall apps.
Restart Manager	The Restart Manager API can eliminate or reduce the number of system restarts that are required to complete an installation or update.
Update Orchestrator	The Update Orchestrator API schedules your background software updates with user impact in mind.
Windows Installer	Microsoft Windows Installer is an installation and configuration service provided with Windows. The installer service enables customers to provide better corporate deployment and provides a standard format for component management. The installer also enables the advertisement of applications and features according to the operating system.
Windows Setup and Migration	This section documents the notifications used to detect and possibly repair an application after a setup or migration has occurred. These notifications can also be used to suspend operations during the volatile setup or migration experience.

Isolated Applications and Side-by-side Assemblies

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Isolated Applications and Side-by-side Assemblies is a Microsoft Windows solution that reduces versioning conflicts in Windows-client applications. With Windows, application developers can build isolated applications that are fully self-describing and unaffected by changes to the registry, other applications, or other versions of assemblies running on the system. Application authors and administrators can use manifests to manage the sharing of side-by-side assemblies after deployment on either a global or per-application basis. Customers benefit from isolated applications that are more stable and more reliably updated.

Where applicable

Isolated applications and side-by-side assembly sharing can be used to develop applications that safely share operating system assemblies. Developers can use this technology to correct DLL versioning conflicts caused by an incompatible version of a shared assembly.

If your application must consistently get the version of a component you have tested, it is possible to isolate your application so that it will always be run with the tested version of the component on the user's computer.

Isolated Applications and Side-by-side Assemblies is intended for the development of desktop style applications.

Developer audience

This documentation is primarily intended for software developers, application developers, and network administrators:

- Software developers who want to create isolated applications that will use the side-by-side assemblies made available by Microsoft and other side-by-side assembly publishers.
- Application developers who are interested in creating their own side-by-side assemblies to isolate their applications.
- Network administrators who want more information about isolated applications.

As the primary reference for side-by-side assembly sharing and isolated applications, this documentation provides general background information about authoring manifests and side-by-side assemblies, installing isolated applications and side-by-side assemblies, and using the Activation Context API.

Run-time requirements

Windows Server 2003 and later or Windows XP and later is required to use side-by-side assemblies and manifests to isolate applications and to use the Activation Context API.

In this section

TOPIC	DESCRIPTION
Reference	Documentation of Isolated Applications and Side-by-side Assemblies.

Packaging, deployment, and query of Windows apps

11/2/2020 • 2 minutes to read • Edit Online

You deploy, manage, and service Windows apps (including UWPs and desktop apps) through .msix/.appx app packages based on the OPC format. Each app package contains the files that constitute the app, and a manifest file that describes the software to Windows.

Introduction

Typically, developers create and sign app packages using Visual Studio. For more info, see Package a UWP app with Visual Studio.

The Microsoft Store makes it easy to build, submit, and sell your apps to customers around the world. For more info, see App submissions.

Windows PowerShell cmdlets enable you to install and manage line-of-business Windows apps without using the Store. For more info, see Appx Module Cmdlets.

Using the packaging, deployment, and query APIs, you can programmatically perform these tasks:

- Create an app package for a Windows app
- Deploy a packaged Windows app
- Enumerate the app packages installed on a system and get information about them from their manifest
- Consume the contents of an app package

In this section

TOPIC	DESCRIPTION
How to create an app package (C++)	Learn how to create an app package using the packaging API.
How to create an app package signing certificate	Learn how to use **MakeCert** and **Pvk2Pfx** to create a test code signing certificate, so that you can sign your app packages.
How to sign an app package using SignTool	Learn how to use **SignTool** to sign your app packages so they can be deployed.
How to troubleshoot app package signature errors	An app deployment failure can be caused by a failure to validate the digital signature of the app package. Learn how to recognize these failures, and what to do about them.
How to programmatically sign an app package (C++)	Learn how to sign an app package by using the **SignerSignEx2** function.
How to develop an OEM app that uses a custom file	Learn how to develop an app that uses a custom file to pass info from the OEM to the app.

TOPIC	DESCRIPTION
Extract app package contents (C++)	Learn how to extract files from an app package using the packaging API.
Query app package manifest info (C++)	Learn how to get info from an app package manifest using the packaging API
Troubleshooting	Provides info to help you troubleshoot problems you experience when packaging, deploying, or querying an app package.
Packaging API reference	The packaging API creates, reads, and writes app packages.
Deployment API reference	The deployment API installs, updates, and uninstalls app packages.
Query API reference	The query API gets info about the app packages installed on the system.
Tools and PowerShell cmdlets	Use these tools and cmdlets to create, install, and manage app packages.
SDK samples	Download SDK samples that demonstrate the packaging, deployment, and query APIs for Windows apps.
Glossary	Learn about the terms related to packaging, deployment, and query of Windows apps.

Related topics

Concepts

App packages and deployment

Other Reference

App package manifest schema

Windows.ApplicationModel.Package

Windows.ApplicationModel.PackageId

Windows.Management.Deployment.PackageManager

Windows.Management.Deployment.PackageUserInformation

Developer licensing

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Developer licensing enables you to get, check, or remove a developer license.

Restart Manager

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Restart Manager API can eliminate or reduce the number of system restarts that are required to complete an installation or update. The primary reason software updates require a system restart during an installation or update is that some of the files that are being updated are currently being used by a running application or service. The Restart Manager enables all but the critical system services to be shut down and restarted. This frees files that are in use and allows installation operations to complete.

Where applicable

The Restart Manager DLL exports a public C interface that can be loaded by standard or custom installers. The installer can use the Restart Manager to register files that should be replaced during the installation of an application or update. Then during a subsequent update or installation, the installer can use the Restart Manager to determine which files cannot be updated because they are currently in use. Restart Manager can shut down and restart the non-critical services or applications that are currently using those files. Installers can direct the Restart Manager to shut down and restart applications or services based on the file in use, the process ID (PID), or the short-name of a Windows service.

Restart Manager is intended for the development of desktop style applications.

Developer audience

This documentation is intended for developers of installation applications who want to take advantage of the installer capabilities in Windows Vista or Windows Server 2008. Applications that use the Windows Installer version 4.0 for installation and servicing automatically use the Restart Manager to reduce system restarts. Custom installers can also be designed to call the Restart Manager API to shut down and restart applications and services. In cases where a system restart is unavoidable, installers can use the Restart Manager API to schedule restarts in such a way that it minimizes the disruption of the user's work flow.

Run-time requirements

The Restart Manager API is available beginning with Windows Vista and Windows Server 2008. Restart Manager consists of a single DLL that applications can load to access the Restart Manager API.

In this section

TOPIC	DESCRIPTION
About Restart Manager	Overview topics that describe the Restart Manager.
Using Restart Manager	Overview topics about using the Restart Manager API.
Restart Manager Reference	Reference topics for the Restart Manager API.

UpdateOrchestrator API

2/18/2021 • 2 minutes to read • Edit Online

UpdateOrchestrator schedules your automatic software updates with user impact in mind. This API allows you schedule automatic download and installs, along with their requirements in order to run updates at an optimal time that minimizes user-present impact. These features particularly benefit lower performance systems with limited or slower computing resources.

Windows 19H1 includes a first-generation solution for automatic software update use cases that were adopted by OS updates and Store App updates and exposes an initial ‘Limited Access’ version of this API for a select set of updaters of 'user-mode' apps as described below.

Features

- Dynamically registers software updaters
- Invokes registered software updaters during optimal times, such as during user absence, to update 'user mode apps'.
  - Includes the ability to 'keep awake' on AC power to further reduce user-away impact.
- Operates on a Trust Model that only invokes trusted 3rd party registered software updaters
  - There are substantial risks when exposing UpdateOrchestrator to any and all callers such as updating BIOS firmware or drivers when a user is not present. UpdateOrchestrator includes a trust model to mitigate these risks.

Developer Audience

IMPORTANT

The UpdateOrchestrator API is currently a limited access feature. This API will become publicly available in a future release.

Use UpdateOrchestrator API if you already have background software updaters for Win32 'user mode' applications such as Adobe's updater for Acrobat Reader or Valve's Steam. This interface is not needed for UWP/Store applications as the Microsoft Store already takes advantage of this functionality for software updates.

To provide the best customer experience, this initial API release is scoped to a select set of registered updaters that meet the following criteria:

- Updates for 'user mode' applications only
- Do not involve BIOS/Firmware/Device or Software Drivers
  - Updating BIOS, firmware, or device/software drivers that have not passed a common quality criteria pose a substantial risk, particularly when a user is not present.
- Participating in the usage of this API entails being able to vouch for all content downloaded and installed by your background software updaters on users systems via audits.

This API may be modified significantly before public release. While UpdateOrchestrator API is under development, this initial release as limited access feature is only for updaters that meet above criteria at this time.

Our aim is to improve the functionality of this API and reduce impact from multiple automatic software updaters on Windows. We would appreciate your input through this **brief survey** to help us understand how UpdateOrchestrator API can better serve your developer needs.

Windows Installer

2/18/2021 • 4 minutes to read • Edit Online

NOTE

This documentation is intended for software developers who want to use Windows Installer to build installer packages for applications. If you're looking for a redistributable for Windows Installer 4.5 and earlier, see this article. Note that there is no redistributable for Windows Installer 5.0. This version is included with the OS in Windows 7, Windows Server 2008 R2, and later client and server releases (including Windows 10).

Microsoft Windows Installer is an installation and configuration service provided with Windows. The installer service enables customers to provide better corporate deployment and provides a standard format for component management. The installer also enables the advertisement of applications and features according to the operating system. For more information, see Platform Support of Advertisement.

This documentation describes Windows Installer 5.0 and earlier versions. Not all the capabilities available in later Windows Installer versions are available in earlier versions. This documentation does not describe versions earlier than Windows Installer 2.0. Installation packages and patches that are created for Windows Installer 2.0 can still be installed by using Windows Installer 3.0 and later.

Windows Installer 3.0 and later, can install multiple patches with a single transaction that integrates installation progress, rollback, and reboots. The installer can apply patches in a specified order regardless of the order that the patches are provided to the system. Patching using Windows Installer 3.0 only updates files affected by the patch and can be significantly faster than earlier installer versions. Patches installed with Windows Installer 3.0 or later can be uninstalled in any order to leave the state of the product the same as if the patch was never installed. Accounts with administrator privileges can use the API of Windows Installer 3.0 and later to query and inventory product, feature, component, and patch information. The installer can be used to read, edit, and replace source lists for network, URL, and media sources. Administrators can enumerate across user and install contexts, and manage source lists from an external process.

Windows Installer 4.5 and later can install multiple installation packages using *transaction processing*. If all the packages in the transaction cannot be installed successfully, or if the user cancels the installation, the Windows Installer can roll back changes and restore the computer to its original state. The installer ensures that all the packages belonging to a multiple-package transaction are installed or none of the packages are installed.

Beginning with Windows Installer 5.0, a package can be authored to secure new accounts, Windows Services, files, folders, and registry keys. The package can specify a security descriptor that denies permissions, specifies inheritance of permissions from a parent resource, or specifies the permissions of a new account. For information, see Securing Resources. The Windows Installer 5.0 service can enumerate all components installed on the computer and obtain the key path for the component. For more information, see Enumerating Components. By Using Services Configuration, Windows Installer 5.0 packages can customize the services on a computer. Setup developers can use Windows Installer 5.0 and Single Package Authoring to develop single installation packages capable of installing an application in either the per-machine or per-user installation context.

Where applicable

Windows Installer enables the efficient installation and configuration of your products and applications running on Windows. The installer provides new capabilities to advertise features without installing them, to install products on demand, and to add user customizations.

Windows Installer 5.0 running on Windows Server 2012 or Windows 8 supports the installation of approved apps on Windows RT. A Windows Installer package, patch, or transform that has not been signed by Microsoft cannot be installed on Windows RT. The **Template Summary** property indicates the platform that is compatible with an installation database and in this case should include the value for Windows RT.

Windows Installer is intended for the development of desktop style applications.

Developer audience

This documentation is intended for software developers who want to make applications that use Windows Installer. It provides general background information about installation packages and the installer service. It contains complete descriptions of the application programming interface and elements of the installer database. This documentation also contains supplemental information for developers who want to use a table editor or a package creation tool to make or maintain an installation.

Run-time requirements

Windows Installer 5.0 is included with, Windows 7, Windows Server 2008 R2, and later releases. There is no redistributable for Windows Installer 5.0.

Versions earlier than Windows Installer 5.0 were released with Windows Server 2008, Windows Vista, Windows Server 2003, Windows XP, and Windows 2000. Windows Installer Redistributables are available for Windows Installer 4.5 and some earlier versions.

- Windows Installer 4.5 requires Windows Server 2008, Windows Vista, Windows XP with Service Pack 2 (SP2) and later, and Windows Server 2003 with Service Pack 1 (SP1) and later.
- Windows Installer 4.0 requires Windows Vista or Windows Server 2008. There is no redistributable for installing Windows Installer 4.0 on other operating systems. An updated version of Windows Installer 4.0, which does not add any new features, is available in Windows Vista with Service Pack 1 (SP1) and Windows Server 2008.
- Windows Installer 3.1 requires Windows Server 2003, Windows XP, or Windows 2000 with Service Pack 3 (SP3).
- Windows Installer 3.0 requires Windows Server 2003, Windows XP, or Windows 2000 with SP3. Windows Installer 3.0 is included in Windows XP with Service Pack 2 (SP2). It is available as a redistributable for Windows 2000 Server with Service Pack 3 (SP3) and Windows 2000 Server with Service Pack 4 (SP4), Windows XP RTM and Windows XP with Service Pack 1 (SP1), and Windows Server 2003 RTM.
- Windows Installer 2.0 is contained in Windows Server 2003 and Windows XP.
- Windows Installer 2.0 is available as a package for installing or upgrading to Windows Installer 2.0 on Windows 2000. This package should not be used to install or upgrade Windows Installer 2.0 on Windows Server 2003 and Windows XP.

In this section

TOPIC	DESCRIPTION
Roadmap	A guide to Windows Installer documentation.
Overview	General information about the installer.
What's New in Windows Installer	Lists additions and changes to Windows Installer.

TOPIC	DESCRIPTION
Reference	Documentation of Windows Installer functions.
Windows Installer Scripting Examples	Windows Installer examples using script.

Audio and Video

2/18/2021 • 2 minutes to read • Edit Online

Microsoft provides components that are designed to enable application developers, Web developers, and systems administrators to develop audio and video programs and to create Windows Media-compatible applications and Web sites.

In this section

TOPIC	DESCRIPTION
Core Audio APIs	A low-level API for audio capture and audio rendering, which can be used to achieve minimum latency or to implement features that might not be entirely supported by higher-level media APIs.
DirectShow	An end-to-end media pipeline, which supports playback, audio/video capture, encoding, DVD navigation and playback, analog television, and MPEG-2.
Microsoft Media Foundation	An end-to-end media pipeline, which supports p playback, audio/video capture, and encoding (successor to DirectShow).
Microsoft TV Technologies	Supports digital television, recorded TV (.wtv) files, and OpenCable Unidirectional Cable Receiver (OCUR) devices.
Windows Media Center Software Development Kit	Using the Windows Media Center Software Development Kit (SDK), developers can create rich media applications and services for use with mouse, keyboard, and remote control on a Media Center PC.
Windows Media Format 11 SDK	Supports reading and writing ASF files, decoding and encoding Windows Media audio and video, and streaming ASF over a network.
Windows Media Library Sharing Services	Enables applications to discover media devices on the home network, and share media libraries on the home network and the Internet.
Windows Media Player SDK	Extends the capabilities of Windows Media Player and Windows Media Player Mobile.
Windows Media Rights Manager 10.1.2 SDK	Supports digital rights management (DRM) for protecting Windows Media files.
Windows Media Services 9 Series	Provides an automation-based API for managing Windows Media Services in Windows Server.
Windows Media Streaming API	Models the Digital Media Renderer (DMR) and Digital Media Server (DMS) devices, as defined by the DLNA guidelines, for easy programmatic use.

TOPIC	DESCRIPTION
Windows Media Transport Controls	Provides a built-in interface that shows the user information about the currently playing media, such as the title of a song or video. Also provides the user with a common and familiar way to control media playback.
Windows Movie Maker 6.0 and Windows DVD Maker 1.0 SDK	Enables customization of Windows Movie Maker 6.0 and Microsoft Windows DVD Maker 1.0.
Legacy Audio and Video	Technologies that are obsolete and should not be used in new applications.

Related topics

DirectX Graphics and Gaming

Graphics

Core Audio APIs

2/18/2021 • 2 minutes to read • Edit Online

This documentation provides information about core audio application programming interfaces (APIs) for the Microsoft Windows family of operating systems. It provides guidelines for software developers to follow in developing applications that use the core audio APIs in Windows Vista. These APIs were new in Windows Vista and are not available in earlier versions of Windows.

The core audio APIs provide the means for audio applications to access audio endpoint devices such as headphones and microphones. The core audio APIs serve as the foundation for higher-level audio APIs such as Microsoft DirectSound and the Windows multimedia **waveXxx** functions. Most applications communicate with the higher-level APIs, but some applications with special requirements might need to communicate directly with the core audio APIs.

Starting with Windows 7, the existing APIs have been improved and new APIs have been added to support new scenarios. The stream and session management APIs have been improved so that the application can now enumerate and get extended control over the audio session. By using the new APIs, the application can implement a custom stream attenuation experience. New device-related APIs provide access to the driver properties of the endpoint devices.

This documentation includes the following sections.

SECTION	DESCRIPTION
About the Windows Core Audio APIs	Provides an overview of the Windows core audio APIs and describes basic concepts.
Programming Guide	Describes the key features of the core audio APIs and how to use them.
Programming Reference	Provides C++ reference information for the core audio APIs.

Related topics

Media Technologies for Windows

DirectShow

2/18/2021 • 2 minutes to read • Edit Online

The Microsoft DirectShow application programming interface (API) is a media-streaming architecture for Microsoft Windows. Using DirectShow, your applications can perform high-quality video and audio playback or capture.

The DirectShow headers, libraries, SDK tools, and samples are available in the Windows SDK.

NOTE

Previous versions of the DirectShow SDK were included in the DirectX SDK. The last version of the DirectX SDK to include DirectShow was DirectX 9.0 SDK Update - (February 2005) Extras. After this version, DirectShow was moved to the Windows SDK. To get the latest version of the DirectShow headers, libraries, and samples, download the Windows SDK.

The DirectShow documentation is divided into the following sections:

- Introduction to DirectShow
- Getting Started
- About DirectShow
- Using DirectShow
- DirectShow Samples
- DirectShow Reference
- DirectX Media Objects
- DirectShow Editing Services
- Appendixes

Related topics

Media Technologies for Windows

Microsoft Media Foundation

11/2/2020 • 2 minutes to read • Edit Online

Microsoft Media Foundation enables the development of applications and components for using digital media on Windows Vista and later.

Media Foundation is the next generation multimedia platform for Windows that enables developers, consumers, and content providers to embrace the new wave of premium content with enhanced robustness, unparalleled quality, and seamless interoperability.

This documentation contains the following sections:

What's New for Media Foundation	Describes what has changed in the latest version of Media Foundation.
About Media Foundation	Lists the headers and libraries you will need, and describes the tools and code samples that are provided to make Media Foundation development easier.
Media Foundation Programming Guide	Describes how to accomplish specific tasks in Media Foundation.
Media Foundation SDK Samples	Describes the code samples for Media Foundation provided in the Windows SDK.
Media Foundation Programming Reference	Contains reference topics for all of the Media Foundation APIs.

Related topics

Media Technologies for Windows

Windows Media Format 11 SDK

11/2/2020 • 4 minutes to read • Edit Online

This documentation describes the Microsoft Windows Media Format Software Development Kit (SDK) and applies to the 32-bit and x64-based versions of the SDK.

The Windows Media Format SDK is a component of the Microsoft Windows Media Software Development Kit (SDK). Other components include the Windows Media Services SDK, Windows Media Encoder SDK, Windows Media Rights Manager SDK, Windows Media Device Manager SDK, and Windows Media Player SDK.

The Windows Media Format SDK provides application developers with access to the components of the Windows Media Format. These components include the Advanced Systems Format (ASF) file container, the Windows Media Audio and Video codecs, basic network streaming capability, and digital rights management. The objects of the Windows Media Format SDK manipulate the components of Windows Media at a low level; the other components of the Windows Media SDK include objects that work on a higher level.

The primary purpose of the Windows Media Format SDK is to enable developers to create applications that play, write, edit, encrypt, and deliver Advanced Systems Format (ASF) files and network streams. These files and streams commonly contain audio and video content encoded using the Windows Media Audio and Video codecs. However, ASF can contain any type of data. For more information about the Advanced Systems Format container structure, see Overview of the ASF Format.

The key features of the Windows Media Format SDK are:

- Support for industry-leading codecs. The Windows Media Format 11 SDK includes the Microsoft Windows Media Video 9 codec and the Microsoft Windows Media Audio 9.1 codec. Both of these codecs provide exceptional encoding of digital media content. New for this release is the Windows Media Video 9 Advanced Profile codec, which provides optimizations for broadcast video. This SDK also includes the Microsoft Windows Media Video 9 Screen codec for compressing computer-screen activity during sessions of user applications, and the Windows Media Audio 9.1 Voice codec, which encodes low-complexity audio such as speech and intelligently adapts to more complex audio such as music, for superior representation of combined voice-music scenarios.
- Support for writing ASF files. Files are created based on customizable profiles, enabling easy configuration and standardization of files. This SDK can be used to write files in excess of 2 gigabytes, enabling longer, better-quality, continuous files.
- Support for reading ASF files. This SDK provides support for reading local ASF files as well as reading ASF data being streamed over a network. Support is also provided for many advanced reading features, such as native support for multiple bit rate (MBR) files, which contain multiple streams with the same content encoded at different bit rates. The reader automatically selects which MBR stream to use, depending upon available bandwidth at the time of playback.
- Support for delivering ASF streams over a network. This SDK provides support for delivering ASF data through HTTP to remote computers on a network, and also for delivering data directly to a remote Windows Media server.
- Support for editing metadata in ASF files. Information about a file and its content is easily manipulated with this SDK. Developers can use the robust system of metadata attributes included in the SDK, or create custom attributes to suit their needs.
- Support for content editing applications. This SDK enables applications to seek to points within a file by presentation time and by video frame. In addition, files created by using the Windows Media Format SDK can maintain timestamps in formats used in film and television production.
- Support for reading and editing metadata in MP3 files. This SDK provides integrated support for reading

MP3 files with the same methods used to read ASF files. Applications built with the Windows Media Format SDK can also edit metadata attributes in MP3 files using built-in support for the most common ID3 tags used by content creators.
- Support for Digital Rights Management protection. This SDK provides methods for reading and writing ASF files and network streams that are protected by Digital Rights Management to prevent unauthorized playback or copying of the content.

To download the Windows Media Format SDK, see the Windows Media Downloads page at the Microsoft Web site.

This document describes how you can develop digital media applications using the Windows Media Format SDK. It is divided into the following sections.

NOTE

Although this document contains information about the latest version of the Windows Media Format SDK, most of the features it describes are supported by older versions of the SDK. Reference pages for the methods, functions, structures, and enumerations of the Windows Media Format SDK include version requirements.

SECTION	DESCRIPTION
About the Windows Media Format SDK	Provides overview and background information that you should be familiar with before attempting to create applications.
Programming Guide	Provides detailed instructions for performing various tasks, such as reading, writing and indexing files, protecting files with Digital Rights Management, streaming ASF data over a network, and so on.
Programming Reference	Provides reference information for the interfaces, methods, functions, structures, enumeration types, and constants related to Windows Media Format.
Windows Media Audio and Video Codec Interfaces	Provides instructions for using the Windows Media Audio and Video codec digital media objects (DMOs) directly.
Glossary	Defines the terms used in the Windows Media Format SDK documentation.

Windows Media Player SDK

11/2/2020 • 2 minutes to read • Edit Online

This documentation describes the Microsoft Windows Media Player Software Development Kit (SDK). The Windows Media Player SDK is one of the components of the Microsoft Windows SDK. Other media components include the Microsoft Media Foundation SDK, the Microsoft Windows Media Format SDK, and the Microsoft Windows Media Services SDK.

The Windows Media Player SDK documents programming technologies that can be used to extend the capabilities of Windows Media Player and Windows Media Player Mobile. These technologies are documented in the following sections:

SECTION	DESCRIPTION
About the Windows Media Player SDK	This section provides details about how to find specific information in the SDK. It includes a section about new features and information about how to use the samples included with the SDK.
Windows Media Player Object Model	The Microsoft Windows Media Player control is a Microsoft ActiveX control used for adding digital media playback capabilities to webpages. It provides a programming interface for rendering digital media files and streams.
Windows Media Player Skins	Skins are an XML-based technology used to customize the user interface of Windows Media Player. You can also use Windows Media Player Mobile skins to customize the user interface of Windows Media Player Mobile.
Windows Media Player Plug-ins	Plug-ins are objects that extend Windows Media Player functionality in a variety of ways. Plug-in types include custom visualizations, user interface plug-ins, DSP plug-ins, and rendering plug-ins. Windows Media Player Mobile plug-in support is also described in this section.
Windows Media Metafiles	Windows Media Metafiles are XML documents that provide information about a media item and its presentation. Metafiles can be used to organize media items into playlists that can include functionality for seamless stream switching, ad insertion, and other features.
Windows Media Playlists	Playlists are files that use XML elements to define either a dynamic "smart" playlist of media items or a static set of media items.
Windows Media Player Online Stores	Windows Media Player provides functionality that enables digital media content providers to integrate their services with Windows Media Player. Integration between the Player and an online digital media store enables the user to locate content, download and manage files, play content, and copy content to CDs or devices.

SECTION	DESCRIPTION
Windows Media Player	Some features of the SDK apply to Windows Media Player, the Windows Media Player ActiveX control, and Windows Media Player Mobile. This section provides information about these features.
Glossary	This section contains definitions of terms used throughout the SDK.

NOTE
Installing the Windows SDK does not install Windows Media Player or Windows Media Player Mobile. You must have Windows Media Player installed to use the material in this SDK. Windows Media Player Mobile is a part of Windows Mobile, which is available only on a Pocket PC or Smartphone supplied by a device manufacturer or mobile operator.

Related topics

Media Technologies for Windows

Windows Media Streaming API

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The Media Streaming API models the Digital Media Renderer (DMR) and Digital Media Server (DMS) devices as defined by the DLNA guidelines for easy programmatic use, enabling developers to create applications with Digital Media Controller (DMC), Digital Media Player (DMP) and Push Controller (+PU+) capabilities as defined by the DLNA guidelines.

In this section

TOPIC	DESCRIPTION
Classes	The Media Streaming API provides the following classes.
Delegates	The Media Streaming API provides the following event handler functions.
Enumerations	The Media Streaming API provides the following enumerations.
Events	The Media Streaming API generates the following events.
Interfaces	The Media Streaming API provides the following interfaces.
Structures	The Media Streaming API provides the following structures.

Developer audience

Media Streaming API is designed for use by C++ developers creating digital media applications that interact with DLNA devices.

Windows Multimedia

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Microsoft Windows multimedia support enables applications to use sound and video.

Where applicable

Windows multimedia can be used in all Windows-based applications.

Developer audience

Windows multimedia is designed for use by C/C++ programmers. Familiarity with the Windows graphical user interface and message-driven architecture is required.

In this section

TOPIC	DESCRIPTION
MCI	General information about how to use the Media Control Interface.
MCIWnd Window Class	General information about a window class that can serve as a control for media playback.
Multimedia Audio	General information about how to use multimedia audio.
Multimedia Input	General information about how to use multimedia input.
Video for Windows	General information about how to use Video for Windows.
Multimedia Reference	Documentation of Windows multimedia functions, interfaces, commands, and messages.

Data Access and Storage

2/18/2021 • 4 minutes to read • Edit Online

Windows has APIs, components, and services that support your desktop apps in data access and storage. They provide:

- File and file system management.
- Database access.
- Support for the transfer, synchronization, and replication of data.
- Access to XML, package, and log files.
- Image mastering.
- Backup support.

In this section

TOPIC	DESCRIPTION
Background Intelligent Transfer Service	Background Intelligent Transfer Service (BITS) transfers files (downloads or uploads) between a client and server and provides progress information related to the transfers. You can also download files from a peer.
Backup	Registry keys for backup and restore allow backup applications to communicate with other applications and services about backup and restore operations. The tape backup API enables backup applications to archive data to tape. The single-instance store (SIS) API enables backup applications to use the SIS architecture for maintaining duplicate files with a minimum of overhead. The raw encryption API enables backup and restore of encrypted files.
Cloud Sync Engines	Starting in Windows 10, version 1709, Windows provides the *cloud files API*. This API formalizes support for cloud sync engines, and handles tasks such as creating and managing placeholder files and directories. Users of this API are typically sync providers and to some extent, Windows applications.
Common Log File System	The Common Log File System (CLFS) API provides a high-performance, general-purpose log file subsystem that dedicated client applications can use and multiple clients can share to optimize log access.
Distributed File System	The Distributed File System (DFS) functions provide the ability to logically group shares on multiple servers and to transparently link shares into a single hierarchical namespace.
Distributed File System Replication	The Distributed File System Replication (DFSR) service is a state-based, multimaster replication engine that supports replication scheduling and bandwidth throttling.

TOPIC	DESCRIPTION
Extensible Storage Engine	The Extensible Storage Engine (ESE) is an advanced indexed and sequential access method (ISAM) storage technology. ESE enables applications to store and retrieve data from tables using indexed or sequential cursor navigation.
File Management API (FMAPI)	The File Management APIs provide a way for developers to discover and restore deleted files from unencrypted volumes. The File Management APIs also provide the ability to use a password or recovery key file for the discovery and recovery of deleted files from BitLocker-encrypted volumes.
Image Mastering API	The image mastering API enables applications to stage and burn images to CD and DVD optical storage media. Other disc-like media that lay images in the same manner can also use this API.
Imaging API	The Windows Imaging Interface Reference describes the programmatic method for managing Windows image (.wim) files.
iSCSI Discovery Library API	The iSCSI Discovery Library API allows initiators to locate any accessible target devices as well as the associated addresses with a minimal amount of required configurations.
iSCSI Software Target API	The iSCSI Software Target API provides a WMI interface for managing Microsoft iSCSI Software Target, such as creating virtual disks and presenting it to the client.
Local File Systems	Describes directory, disk, file, and volume management. Also describes Transactional NTFS (TxF).
MSXML	Microsoft XML Core Services (MSXML) allows customers who use JScript, Visual Basic Scripting Edition (VBScript), and Microsoft Visual Studio to build high-performance XML-based applications.
Non-Volatile Memory Library (NVML)	Allows developers to utilize NVML APIs in order to code for persistent memory in Windows environments.
Offline Files	The Offline Files API allows applications to control and monitor the behavior of Offline Files programmatically.
Packaging	The packaging APIs provide support for applications that produce or consume files, called packages, that are compliant with the Open Packaging Conventions.
Projected File System	The Projected File System (ProjFS) allows a user-mode application to project a hierarchical data store into the file system, where it appears as files and directories. Content is cached to the local file system on demand, allowing very large data stores to appear local without overwhelming local storage.
Remote Differential Compression	Remote Differential Compression (RDC) allows applications to synchronize data between two computers in an efficient manner.

TOPIC	DESCRIPTION
User State Management API	The User State Management API provides an alternative way to configure and retrieve current status for the Windows components related to user state. The Windows components that expose configuration and status through these APIs are Folder Redirection, Offline Files, and Roaming Profiles.
Virtual Disk Service	The Virtual Disk Service (VDS) manages a wide range of storage configurations, from single-disk desktops to external storage arrays.
Virtual Storage	The Virtual Hard Disk (VHD) format is a publicly-available image format specification that specifies a virtual hard disk encapsulated in a single file, capable of hosting native file systems while supporting standard disk and file operations.
Volume Shadow Copy Service	The Volume Shadow Copy Service (VSS) is a set of COM interfaces that implements a framework to allow volume backups to be performed while applications on a system continue to write to the volumes.
Windows Data Access Components	Windows Data Access Components (Windows DAC) 6.0 is a set of technologies that provide access to information across the enterprise. These technologies include Microsoft ActiveX Data Objects (ADO), OLE DB, and Microsoft Open Database Connectivity (ODBC).
Windows Storage Management API	The Windows Storage Management API is used to manage a wide range of storage configurations, from single-disk desktops to external storage arrays.
Windows Sync	The Microsoft Windows Sync API provides a way for developers to write custom synchronization providers that enable devices to synchronize data with data stores on a computer or on a network.
WMI Provider for NFS	Microsoft Services for Network File System (NFS) provides a file sharing solution that enables you to transfer files using the NFS protocol between computers running Windows and third-party operating systems.
XmlLite	XmlLite is a lightweight XML parser designed for ease of use, performance, and standards compliance.

Background Intelligent Transfer Service

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Background Intelligent Transfer Service (BITS) is used by programmers and system administrators to download files from or upload files to HTTP web servers and SMB file shares. BITS will take the cost of the transfer into consideration, as well as the network usage so that the user's foreground work has as little impact as possible. BITS also handles network interuptions, pausing and automatically resuming transfers, even after a reboot. BITS includes PowerShell cmdlets for creating and managing transfers as well as the BitsAdmin command-line utility.

NOTE

BITS can be used by Windows to download updates to your local system. If you are an end-user searching for ways to troubleshoot your BITS installation, see Fix Windows Update Issues.

Where applicable

Use BITS for applications that need to:

- Download from or upload files to an HTTP or REST web server or SMB file server.
- Automatically resume file transfers after network disconnects and computer restarts.
- Preserve the responsiveness of other network applications.
- Be mindful of the network cost on e.g. roaming networks
- Optionally work with BranchCache to optimize wide area network (WAN) traffic

Developer audience

BITS is a COM interface designed for C and C++ developers that can also be used by .NET developers. UWP developers should use the Windows.Networking.BackgroundTransfer API and not the BITS API.

BITS versions

For complete version history and information on earlier operating system, see What's New.

In this section

TOPIC	DESCRIPTION
About BITS	General information about BITS.
Using BITS	Procedural guide for developing BITS clients that transfer files between a client and server.
BITS Reference	Reference information for the BITS programming interfaces. Also contains information about samples, tools, server settings for upload jobs, and the upload protocol.

TOPIC	DESCRIPTION
Best Practices	Information to consider when designing an application that uses BITS.

Additional resources

The following are additional resources.

.NET Reference DLL	For information on using BITS from .NET using reference DLLs, see Calling into BITS from .NET using Reference DLLs
.NET Wrapper	For other .NET wrappers for BITS, you can search nuget for projects tagged with the BITS tag.

Backup

11/2/2020 • 2 minutes to read • Edit Online

Registry keys for backup and restore allow backup applications to communicate with other applications and services about backup and restore operations. For more information, see Registry Keys and Values for Backup and Restore.

The tape backup API enables backup applications to read from and write to a tape, initialize a tape, and retrieve tape or tape drive information. For more information, see Tape Backup.

Single-instance store (SIS) is an architecture designed to maintain duplicate files with a minimum of overhead. Backup application use the SIS backup API to access the SIS architecture. For more information, see Single-Instance Store and SIS Backup.

Backup and restore of encrypted files is enabled by the raw encryption API, which reads and writes encrypted files while keeping the data in encrypted format. The API allows the encrypted data in these files to be backed up and restored, while meeting the goals of maintaining the security of the backed up data, and being usable by an application that lacks permission to use the encryption keys on the file. For more information, see Backup and Restore of Encrypted Files.

The Srdelayed tool can enable system state recovery applications to move, delete, and set the short name of system files. For more information, see Srdelayed.exe.

Related topics

Backup Reference

Cloud Sync Engines

2/18/2021 • 2 minutes to read • Edit Online

Starting in Windows 10, version 1709, Windows provides the *cloud files API*. This API consists of several native Win32 and WinRT APIs that formalize support for cloud sync engines, and handles tasks such as creating and managing placeholder files and directories. Users of this API are typically sync providers and to some extent, Windows applications.

In this section

TOPIC	DESCRIPTION
Build a Cloud Sync Engine that Supports Placeholder Files	Learn how to use the cloud files API to build a cloud file sync engine that includes placeholder files and File Explorer integration.
Cloud Filter Reference	The cloud filter API is a native Win32 API that is part of the broader cloud files API. The cloud filter API provides functionality at the boundary between the user mode and the file system. This API handles the creation and management of placeholder files and directories.

Extensible Storage Engine

2/18/2021 • 2 minutes to read • Edit Online

***Applies to:** Windows | Windows Server*

Extensible Storage Engine

The Extensible Storage Engine (ESE) is an advanced indexed and sequential access method (ISAM) storage technology. ESE enables applications to store and retrieve data from tables using indexed or sequential cursor navigation. It supports denormalized schemas including wide tables with numerous sparse columns, multi-valued columns, and sparse and rich indexes. It enables applications to enjoy a consistent data state using transacted data update and retrieval. A crash recovery mechanism is provided so that data consistency is maintained even in the event of a system crash. It provides ACID (Atomic Consistent Isolated Durable) transactions over data and schema by way of a write-ahead log and a snapshot isolation model. Transactions in ESE are highly concurrent, making ESE useful for server applications. It caches data to maximize high performance access to data. In addition, it is lightweight, making it useful for applications that serve in auxiliary roles.

ESE is for use in applications that require fast and/or light structured data storage, where raw file access or the registry does not support the application's indexing or data size requirements.

It is used by applications that never store more than 1 megabyte of data, and has been used in applications with databases in extreme cases in excess of 1 terabyte, and commonly over 50 gigabytes.

This documentation is intended for developers who are familiar with C and C++, and basic database concepts such as tables, columns, indexes, recovery, and transactions. The only access method for ESE is the C API that is described in this documentation.

The Extensible Storage Engine is a Windows component that was introduced in Windows 2000. Not all features or APIs are available in all versions of the Windows operating systems.

ESE provides a user-mode storage engine that manages data inside of flat, binary files that are accessible through the Windows APIs. ESE is accessed through a DLL that is loaded directly into the application's process; no remote access methods are required of or provided by the database engine itself. Though ESE has no remote or inter-process access method, the data files it uses can be provided remotely by using server message block (SMB) through the Windows APIs, but this is not recommended.

Note Windows XP 64-Bit Edition is the same as Windows Server 2003 for the purpose of determining the ESE feature set that is supported.

Notes

ESE was formerly known as Joint Engine Technology (JET) Blue, and so frequently the term "JET Blue" or "JET" is used interchangeably with the term ESE outside this documentation. However, there are in fact two completely separate implementations of the JET API, called JET Blue and JET Red. The term "JET" is frequently also used to refer to JET Red, which is the database engine that is used with Microsoft Office Access. The two JET implementations are completely different, are separately maintained, have a vastly different feature set, and are not interchangeable. Within the ESE documentation, "JET" refers to the ESE or the JET API as ESE implements it. Any references to the JET Red will always explicitly be labeled "JET Red".

In This Section

Extensible Storage Engine Reference

Image Mastering API

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Microsoft Windows image mastering API enables applications to stage and burn images to CD, DVD, and Blu-ray optical storage media. Other disc-like media that lay images in the same manner can also use this API.

Developer audience

IMAPI is designed for C/C++ and Visual Basic developers and those writing scripts using VBScript.

Knowledge of CD, DVD, and Blu-ray technologies and file systems is recommended. Developers may benefit from a familiarity with the latest revision of Multimedia Command (MMC) at ftp://ftp.t10.org/t10/drafts/mmc6.

Run-time requirements

IMAPI 2.0 is supported natively starting with Windows Vista and Windows Server 2008. Enabling IMAPI 2.0 functionality for Windows XP and Windows Server 2003 requires the installation of the KB932716 update package. If this package is not installed, Windows XP still natively supports IMAPI 1.0.

NOTE
The Windows Feature Pack for Storage is available to the public via the Microsoft Download Center. Additional information regarding specific API changes can be found in the What's New section.

In this section

TOPIC	DESCRIPTION
What's New	Information detailing each significant release of the Image Mastering API.
About IMAPI	General information about the Image Mastering API.
Using IMAPI	Task-related topics that demonstrate how to use the Image Mastering API.
IMAPI Reference	Detailed descriptions of IMAPI interfaces and other programming elements.

Additional resources

Further information regarding IMAPI and other related subjects can be found at the following locations:

Microsoft Optical Storage Blog	Highlights topics that focus on the implementation of the Image Mastering API in real world development scenarios.

Optical Platform Discussion Forum	Discuss CDROM.SYS, IMAPIv2 and Live File System issues. This forum focuses on system-level topics and is intended for application developers rather than endusers.
T10 Technical Committee	A Technical Committee of the InterNational Committee on Information Technology Standards (INCITS, pronounced "insights"). INCITS is accredited by, and operates under rules that are approved by, the American National Standards Institute (ANSI). These rules are designed to ensure that voluntary standards are developed by the consensus of industry groups.
Optical Storage Technology Association (OSTA)	Works to shape the future of the industry through regular meetings of Commercial Optical Storage Applications (COSA), DVD Compatibility, Marketing, MPV (MusicPhotoVideo), UDF committees, and a new adhoc Blue Laser committee. Interested companies worldwide are invited to join the organization and participate in its committees and programs.

Local File Systems

2/18/2021 • 2 minutes to read • Edit Online

A *file system* enables applications to store and retrieve files on storage devices. Files are placed in a hierarchical structure. The file system specifies naming conventions for files and the format for specifying the path to a file in the tree structure.

Each file system consists of one or more drivers and dynamic-link libraries that define the data formats and features of the file system. File systems can exist on many different types of storage devices, including hard disks, jukeboxes, removable optical disks, tape back-up units, and memory cards.

All file systems supported by Windows have the following storage components:

- Volumes. A *volume* is a collection of directories and files.
- Directories. A *directory* is a hierarchical collection of directories and files.
- Files. A *file* is a logical grouping of related data.

In this section

TOPIC	DESCRIPTION
Directory Management	A *directory* is a hierarchical collection of directories and files. The only constraint on the number of files that can be contained in a single directory is the physical size of the disk on which the directory is located.
Disk Management	A *hard disk* is a rigid disk inside a computer that stores and provides relatively quick access to large amounts of data. It is the type of storage most often used with Windows. The system also supports removable media.
File Management	A *file object* provides a representation of a resource (either a physical device or a resource located on a physical device) that can be managed by the I/O system.
Transactional NTFS (TxF)	Transactional NTFS (TxF) allows file operations on an NTFS file system volume to be performed in a transaction.
Volume Management	The highest level of organization in the file system is the *volume*. A file system resides on a volume.

Related topics

FAT Technical Reference

File Systems Technologies

NTFS Technical Reference

Windows Projected File System (ProjFS)

11/2/2020 • 2 minutes to read • Edit Online

The Windows Projected File System (ProjFS) allows a user-mode application called a "provider" to project hierarchical data from a backing data store into the file system, making it appear as files and directories in the file system. For example, a simple provider could project the Windows registry into the file system, making registry keys and values appear as files and directories, respectively. An example of a more complex provider is VFS for Git, which is used to virtualize very large git repos.

NOTE

ProjFS is designed for use with high-speed backing data stores. One of its design goals is to make the projected data appear as if it were locally present, hiding the fact that the data may be remote. As such, ProjFS doesn't provide: mechanisms for reporting progress of data recall; indication of the online versus offline state of a file; nor other features that may be desirable when working with backing data stores that are slow. For such scenarios, consider instead using the Cloud Files API.

In this section

TOPIC	DESCRIPTION
Windows Projected File System Programming Guide	Conceptual information on implementing a ProjFS provider application.
Windows Projected File System API Reference	Reference information for the ProjFS programming interface.
Windows Projected File System glossary	Special terms used in ProjFS.

Additional Resources

RegFS Sample	A sample ProjFS provider that projects the Windows registry into the file system.

Persistent Memory Programming in Windows - NVML Integration

2/18/2021 • 2 minutes to read • Edit Online

Persistent memory (PM) technology provides byte-level access to non-volatile media while also reducing the latency of storing or retrieving data significantly. It creates a new tier between a system's memory and traditional storage. Any program that is dependent on or scales with quick writes to a persistent medium can benefit from PM.

The purpose of this article is to outline how the non-volatile memory library (NVML) can be integrated into a Visual Studio project for easy use.

NOTE

Persistent Memory is sometimes also referred to as Storage Class Memory (SCM).

PM and NVML

First support for persistent memory was introduced in Windows Server 2016 and the Windows 10 Anniversary Update (1607). For a quick overview, check out these two Channel9 videos:

- Using Non-volatile Memory (NVDIMM-N) as Block Storage in Windows Server 2016
- Using Non-volatile Memory (NVDIMM-N) as Byte-Addressable Storage in Windows Server 2016

To help developers take advantage of the benefits persistent memory offers, Microsoft has also contributed to the efforts of bringing the non-volatile memory library (NVML) to Windows. This library provides various tools to make applications persistent-memory aware. For example, it contains code that lets you easily create a PM-aware key-value store for extremely fast look-ups and stores. You can find more information on NVML, including samples, at NVM Library.

Integrating NVML into a Visual Studio Project

1. Download NVML library files and headers

- NVML is maintained on GitHub. You can either compile the source yourself, or just download compiled binaries directly from here: NVML Version 1.2 - Windows Technical Preview.

2. Place the library files and headers in a directory of your choosing, for example: "C:\NVML\lib" and "C:\NVML\inc" respectively.

3. Configure your project as follows:

- Open your visual studio project and in the "Solution Explorer" right-click on your project's name.
- Open the project's setting pane at the bottom of the resulting pop-up.
- Navigate to "Configuration Properties -> C/C++" and add the folder in which you stored the header (C:\NVML\inc) to the "Additional Include Directories" field.
- Next, navigate to "Configuration Properties -> Linker" and add the folder in which you stored the library (C:\NVML\lib) to the "Additional Library Directories" field

4. Next, make sure you target the project for Windows Server 2016 or Windows 10 Anniversary Update:

- Navigate to “Configuration Properties -> General” and set the “Target Platform Version” field to “10.0.14393.0” and
- Navigate to “Configuration Properties -> C/C++” and add “NTDDI_VERSION=NTDDI_WIN10_RS1;” to the “Preprocessor” field.

5. Include the headers in your code and link to the required libraries

- At this point, you can simply include the header files you are looking to use in your code like any other header files. For example, to use libpmem:
  - add "#include <libpmem.h>" and
  - add "libpmem.lib" to "Configuration Properties -> Linker -> Input -> Additional Dependencies"

At this point you are ready to call the library’s functions directly in your code and take advantage of them.

Volume Shadow Copy Service

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Volume Shadow Copy Service (VSS) is a set of COM interfaces that implements a framework to allow volume backups to be performed while applications on a system continue to write to the volumes.

For an introduction to VSS for system administrators, see Volume Shadow Copy Service in the TechNet Library.

Run-time requirements

VSS is supported on Microsoft Windows XP and later. For information about run-time requirements for a particular programming element, see the Requirements section of the documentation for that element.

All 32-bit VSS applications (requesters, providers, and writers) must run as native 32-bit or 64-bit applications. Running them under WOW64 is not supported. For more information, see Supported Configurations and Restrictions.

Windows Server 2003 and Windows XP: Running 32-bit VSS requesters under WOW64 is supported, but not for system-state backups. Running 32-bit VSS providers and writers under WOW64 is not supported. Support for running 32-bit requesters under WOW64 was removed in Windows Vista and subsequent versions.

NOTE

A shadow copy that was created on Windows Server 2003 R2 or Windows Server 2003 cannot be used on a computer that is running Windows Server 2008 R2 or Windows Server 2008. A shadow copy that was created on Windows Server 2008 R2 or Windows Server 2008 cannot be used on a computer that is running Windows Server 2003. However, a shadow copy that was created on Windows Server 2008 can be used on a computer that is running Windows Server 2008 R2, and vice versa.

In this section

TOPIC	DESCRIPTION
Overview	Describes the VSS object model, backup and restore strategies, and how to create VSS providers, requesters, and writers.
Reference	Describes VSS classes, data types, enumerations, functions, interfaces, and structures.

Additional resources

Windows Vista and later	VSS is available in the Microsoft Windows Software Development Kit (SDK). You can install the SDK for Windows 7 and Windows Server 2008 R2 from the Windows Download Center. You can also download the ISO version of the SDK from the Windows Download Center. Previous versions of the SDK can be downloaded from the Windows SDK Download Page.
Windows Server 2003 and Windows XP	VSS is available in the Volume Shadow Copy Service 7.2 SDK, which you can download from the Windows Download Center. Note that the 64-bit vssapi.lib files in the directories under the Win2003\Obj directory can be used for the 64-bit versions of Windows Server 2003 and Windows XP.

Devices

2/18/2021 • 2 minutes to read • Edit Online

In this section

TOPIC	DESCRIPTION
Device Access	The Device Access APIs enable developers to write Windows Store apps for custom devices. The API applies a policy to grant a Windows Store app the appropriate level of access to a device, and if access is granted, provides methods for sending control codes to operate the device.
AllJoyn API	AllJoyn is a DCOM-like framework and protocol for making remotable method calls and/or sending one-way signals between applications on a distributed bus. It is intended to be used primarily for Internet of Things type scenarios.
Point of Service API	The Point of Service API enables application developers to access Point of Service (POS) peripheral devices. The namespace provides a vendor-neutral interface for accessing POS devices from a UWP app.
Communications Resources	A communications resource is a physical or logical device that provides a single bidirectional, asynchronous data stream. Serial ports, parallel ports, fax machines, and modems are examples of communications resources. For each communications resource, there is a service provider, consisting of a library or driver, that enables applications to access the resource.
Device Management	Device management provides a way to uniformly notify all applications and system components of changes that may affect their operation or access to resources. Applications and the system use and process device events to take advantage of new resources when they become available and to prevent loss of data when existing resources become unavailable.
Enhanced Storage	The Enhanced Storage API enables consistent end-to-end authentication for personal storage devices that differ in form factor.
Function Discovery	Function Discovery provides a uniform programmatic interface for enumerating system resources, such as hardware devices, whether they are local or connected through a network.
PnP-X	Plug and Play Extensions (PnP-X) enable a computer system to discover networked devices and to install them on the local system using Plug and Play (PnP).

TOPIC	DESCRIPTION
Location API	The Location API helps to simplify location-aware programming by providing a standard way to retrieve data about user location and standardizing formats for location data reports. The Location API automatically handles transitions between location data providers and always chooses the most accurate provider for the current situation.
Sensor API	The Sensor API enables applications to get and use data from sensors in a standardized way.
UPnP APIs	The UPnP framework enables dynamic networking of intelligent appliances, wireless devices, and PCs.
Web Services on Devices	The Microsoft Web Services on Devices API (WSDAPI) supports the implementation of client-controlled devices and services, and device hosts conforming to the Devices Profile for Web Services (DPWS).
Windows Media Device Manager 11 SDK	Applications or components built on Windows Media Device Manager have a consistent API for communicating with a wide range of devices including Media Transfer Protocol (MTP), Mass Storage Class (MSC), RAPI, and other devices built on both the latest and previous versions of Windows Media technology.
Windows Mixed Reality	Windows Mixed Reality enables the building of mixed reality experience for Microsoft HoloLens and other immersive headsets. For more information, see the Mixed Reality Dev Center.
Windows Portable Devices	Windows Portable Devices (WPD) enables computers to communicate with attached media and storage devices. WPD provides a flexible, robust way for computers to communicate with music players, storage devices, mobile phones, cameras, and many other types of connected devices.

Communications Resources

2/18/2021 • 2 minutes to read • Edit Online

A *communications resource* is a physical or logical device that provides a single bidirectional, asynchronous data stream. Serial ports, parallel ports, fax machines, and modems are examples of communications resources. For each communications resource, there is a service provider, consisting of a library or driver, that enables applications to access the resource.

- About Communications Resources
- Using Communications Resources
- Communications Reference

Device Access API

11/2/2020 • 2 minutes to read • Edit Online

Purpose

You can use the Device Access API to write Windows Store device apps for specialized devices that use custom drivers. The API provides methods for sending control codes to communicate with the device's custom driver.

In this section

TOPIC	DESCRIPTION
About the Device Access API	The Device Access API is for C++ developers who are creating a Windows Store app to interact with specialized devices in Windows 8. This topic describes the scenarios that the Device Access API applies to. It also explains how the Device Access API applies security rules for Windows Store apps in Windows 8.
How to Use the Device Access API	This topic contains tasks and design considerations for using the Device Access API.
Device Access API C++ Programming Reference	Provides reference pages for the functions and interfaces in the Device Access API.
Device Access Glossary	The following are terms used throughout the documentation for the Device Access API.
Other APIs	These interfaces are not supported and should not be used. Use the APIs in the Device Access API C++ Programming Reference instead.

Developer audience

The Device Access API is designed for independent hardware vendor (IHV) and OEM developers who are familiar with C++ and Component Object Model (COM).

Related topics

Custom Driver Access Sample, UWP device apps for internal devices, Hardware Dev Center

Location API

2/18/2021 • 2 minutes to read • Edit Online

[The Win32 Location API and is available for use in the operating systems specified in the Requirements section. It may be altered or unavailable in subsequent versions. Instead, use the **Windows.Devices.Geolocation** API. To access location from a website, use the W3C Geolocation API.]

Purpose

Computers today are more mobile than ever. From small laptops to Tablet PCs, many computers can go wherever the user wants to go. Programs that take advantage of the computer's mobility can add significant value to people's lives. For example, a program that can find nearby restaurants and provide driving directions would seem to be a natural fit for a portable computer. But while the technology to determine the user's current location is common and affordable, building solutions on this technology can be a daunting task.

To create a location-aware program, you might need to overcome a variety of issues, including:

- Global positioning system (GPS) devices that use virtual COM ports, which provide access for only one program at a time.
- Understanding and programming for protocols, such as the National Marine Electronics Association (NMEA) specification, as well as proprietary vendor extensions.
- Being confined to programming for known, vertical hardware solutions.
- Implementing logic to handle transitions between various location providers, such as GPS receivers, connected networks, cellular telephone networks, the Internet, and user settings.

This documentation describes the Windows Location application programming interface (API). The Location API helps to simplify location-aware programming by providing a standard way to retrieve data about user location and standardizing formats for location data reports. The Location API automatically handles transitions between location data providers and always chooses the most accurate provider for the current situation.

Developer audience

The Location API provides its functionality through a set of COM interfaces. Location API functionality can be used by programmers who are familiar with using COM through the C++ programming language, or with using COM objects in scripting languages, such as Microsoft JScript.

In this section

- Location API C++ Programming Reference
- Location API Object Model Reference

Sensor API

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Windows 7 includes native support for sensors, which are devices that can measure physical phenomena such as temperature or location. This documentation describes the Sensor API, which enables applications to get and use data from sensors in a standardized way.

As humans, we rely on our senses to provide us with information about the world around us. When we create machines to take on some of our work, we add sensor mechanisms so the machines can respond appropriately to changing conditions.

For example, automobile engines typically use a variety of sensors. These sensors are monitored by an onboard computer that continuously adjusts settings, such as engine timing, to maximize power and efficiency. A television may use an ambient light sensor to adjust the brightness of the picture to match changing room conditions. Even something as simple as a doorbell button acts as a rudimentary sensor to detect a human presence at the door.

While the purely mechanical doorbell fulfills its purpose, the information provided by complex sensors becomes far more powerful when it is combined with software. Modern sensors can provide a lot of data very quickly, and in a variety of formats, so software provides a natural mechanism for making sense of sensor data.

Today, software developers can write programs that use sensors, but a lack of standardization makes programming for sensors an arduous task. After a sensor-based program is completed, it is usually forever dependent on a particular type of hardware. Using one or more vertical solutions to enable deployment of a software-based system has limited the integration of sensors with computer hardware and, until now, Windows-based computers have been no exception.

Windows 7 includes native support for sensors, expanded by a new development platform for working with sensors, including location sensors, such as GPS devices. The Windows Sensor and Location platform provides a standard way for device manufacturers to expose sensor devices to software developers and consumers, while providing developers with a standardized application programming interface (API) for working with sensors and sensor data.

Sensors are devices or mechanisms that can measure physical phenomena, provide descriptive data, or provide information about the state of a physical object or environment. Computers can make use of built-in sensors, sensors that are connected through wired or wireless connections, or sensors that provide data through a network or the Internet.

The Sensor API provides a standard way to programmatically access data that sensors provide. The Sensor API standardizes:

- Sensor categories, types, and properties.
- Data formats for standard sensor types.
- COM interfaces for working with sensors and collections of sensors.
- Event mechanisms for asynchronously receiving sensor data.

The Sensor API also enables you to define custom sensor categories, types, properties, data formats, and events.

Developer audience

The Sensor API provides its functionality through a set of COM interfaces. This documentation assumes that you have a working knowledge of programming using the C++ programming language, and you have a basic understanding of how to use COM objects and interfaces. For the sake of brevity, many code examples in this documentation (as well as in the code samples) use Active Template Library (ATL) objects to implement COM functionality.

In this section

- Getting Started
- About the Sensor API
- Sensor API Programming Guide
- Sensor API Programming Reference

UPnP APIs

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The UPnP framework enables dynamic networking of intelligent appliances, wireless devices, and PCs. There are two APIs for working with UPnP-certified devices:

- The Control Point API, which consists of a set of COM interfaces used to find and control devices.
- The Device Host API, which consists of a set of COM interfaces used to implement devices that are hosted by a computer.

Where applicable

The Control Point API enables developers to write applications that search for and control UPnP-certified devices. The Device Host API enables developers to implement the functionality of UPnP-certified devices, and use the device host to manage the discovery, description, control, presentation, and eventing functions of a UPnP-certified device.

Developer audience

Developers using the Control Point APIs and Device Host APIs must be familiar with the UPnP device architecture. For more information, see the UPnP Implementation Documentation and the UPnP Forum.

Developers who are using the Device Host APIs should be familiar with the Active Template Library (ATL) and COM interfaces.

The Control Point APIs and Device Host APIs are used by a variety of applications, from scripts embedded in HTML pages to full-fledged C++ and Microsoft Visual Basic programs.

Only the Control Point API supports Visual Basic Scripting Edition (VBScript).

Run-time requirements

The Control Point API is used on computers running Microsoft Windows Millennium Edition, Windows XP, Windows XP Professional, and Windows CE .NET.

The Device Host API is used on computers running Windows XP, Windows XP Professional, and Windows CE .NET.

For more specific information about which operating systems support a particular function, refer to "Requirements" in the documentation.

In this section

TOPIC	DESCRIPTION
Overview of UPnP Architecture	General information and background.
Control Point Overview	General information about the Control Point API.

TOPIC	DESCRIPTION
Using the Control Point API	Sample code that shows how to develop applications that control UPnP-certified devices.
Control Point API Reference	Documentation of Control Point component interfaces, methods, and events.
Device Host API Overview	General information about the Device Host API.
Using the Device Host API	Sample code that shows how to develop an application for UPnP-certified devices.
Device Host API Reference	Documentation of Device Host component interfaces, methods, and events.

Web Services on Devices

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Microsoft Web Services on Devices API (WSDAPI) supports the implementation of client-controlled devices and services, and device hosts conforming to the Devices Profile for Web Services (DPWS). WSDAPI uses WS-Discovery for device discovery.

WSDAPI may be used for the development of both client and service implementations.

Where applicable

Web Services on Devices allows a client to discover and access a remote device and its associated services across a network. It supports device discovery, description, control, and eventing. Developers can create WSDAPI client proxies and corresponding stubs for device hosts.

Developer audience

The Web Services on Devices documentation is intended for C/C++ programmers and device vendors creating DPWS-compliant products.

Run-time requirements

Client applications that use WSDAPI are supported starting with Windows Vista and Windows Server 2008.

In this section

TOPIC	DESCRIPTION
About Web Services on Devices	Architectural and general information about Web Services on Devices.
Using Web Services on Devices	Information about generating code, configuring applications, and troubleshooting.
Web Services on Devices Reference	Reference documentation for the Web Services on Devices API.

Related topics

Dan Driscoll's Blog

Web Services on Devices

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Microsoft Web Services on Devices API (WSDAPI) supports the implementation of client-controlled devices and services, and device hosts conforming to the Devices Profile for Web Services (DPWS). WSDAPI uses WS-Discovery for device discovery.

WSDAPI may be used for the development of both client and service implementations.

Where applicable

Web Services on Devices allows a client to discover and access a remote device and its associated services across a network. It supports device discovery, description, control, and eventing. Developers can create WSDAPI client proxies and corresponding stubs for device hosts.

Developer audience

The Web Services on Devices documentation is intended for C/C++ programmers and device vendors creating DPWS-compliant products.

Run-time requirements

Client applications that use WSDAPI are supported starting with Windows Vista and Windows Server 2008.

In this section

TOPIC	DESCRIPTION
About Web Services on Devices	Architectural and general information about Web Services on Devices.
Using Web Services on Devices	Information about generating code, configuring applications, and troubleshooting.
Web Services on Devices Reference	Reference documentation for the Web Services on Devices API.

Related topics

Dan Driscoll's Blog

Windows Media Device Manager 11 SDK

11/2/2020 • 2 minutes to read • Edit Online

IMPORTANT

The Windows Media Device Manager APIs are now included in the Windows SDK. No additional SDK downloads are required.

The Microsoft Windows Media Device Manager Software Development Kit (SDK) enables you to build desktop applications and components that can communicate with connected portable devices. Windows Media Device Manager enables your application or component to enumerate, explore, and exchange files with a device, query for metadata, and request play count information. Applications or components built on Windows Media Device Manager have a consistent API for communicating with a wide range of devices including Media Transfer Protocol (MTP), Mass Storage Class (MSC), RAPI, and other devices built on both the latest and previous versions of Windows Media technology.

You can build the following items using the Windows Media Device Manager SDK:

- Desktop applications, such as custom media players. All communication between an application and a portable device must go through Windows Media Device Manager, which acts as a broker between the application, the Microsoft digital rights management system, and the service provider.
- Service providers, which act as the communication link between a custom device and a desktop application. Although Microsoft provides a number of service providers that can communicate with most devices out of the box, you can build a custom service provider to customize the communication between a device and a desktop application.
- Plug-ins, which can perform tasks such as requesting and logging play counts on devices. These plug-ins can be attached to other desktop applications, such as the Windows Media Player to send information to content providers to handle royalty payments to artists.

To download the Windows Media Device Manager SDK, see the Windows Media Download page on the MSDN Web site.

This SDK is a component of the Microsoft Windows Media Software Development Kit. Other components include the Windows Media Format SDK, the Windows Media Services SDK, the Windows Media Encoder SDK, the Windows Media Rights Manager SDK, and the Windows Media Player SDK.

This documentation includes the following sections.

SECTION	DESCRIPTION
Getting Started	Describes what is new in this version of Windows Media Device Manager, gives an overview of how Windows Media Device Manager works, describes what will be needed to develop an application or service provider, and explains the samples shipped with the SDK.
Programming Guide	Discusses how to build applications and service providers.

SECTION	DESCRIPTION
Programming Reference	Provides C++ reference information for the interfaces, methods, classes, and structures supported by the Windows Media Device Manager SDK.
Glossary	Defines terms used in this documentation.

Windows Portable Devices

2/18/2021 • 2 minutes to read • Edit Online

Windows Portable Devices (WPD) enables computers to communicate with attached media and storage devices. WPD provides a flexible, robust way for computers to communicate with music players, storage devices, mobile phones, cameras, and many other types of connected devices. This system supersedes both Windows Media Device Manager and Windows Image Acquisition.

Applications that are built on WPD can explore a device, send and receive content, and even control the device, for example, take a picture or send a text message. The system is designed to be flexible so that many types of devices can be explored, and extensible so that driver developers can define custom properties and commands for custom devices.

You can write both Windows applications and Web applications with WPD. You use the WPD Application Programming Interface to create Windows applications. These applications can be written in C++, C# .Net, or Visual Basic .Net. You use the WPD Automation Object Model to write web applications. These applications are written in JScript and HTML.

The WPD Application Programming Interface is supported in Windows 7, Windows Vista, and Windows XP operating systems. The WPD Automation Object Model is only supported in Windows 7.

For more information about creating a WPD Windows application, refer to the WPD Application Programming Interface documentation and samples.

For more information about creating a WPD Automation application, refer to the WPD Automation Object Model documentation and samples.

Related topics

WPD Application Programming Interface

WPD Automation Object Model

WPD Application Programming Interface

2/18/2021 • 2 minutes to read • Edit Online

Applications built on Windows Portable Devices can explore a device, send and receive content, and even control the device, for example, take a picture or send a text message. The system is designed to be flexible so that many types of devices can be explored, and extensible so that driver developers can define custom properties and commands for custom devices.

The following table describes the main topics of this documentation.

TOPIC	DESCRIPTION
General Requirements for Application Development	Hardware and software requirements to develop drivers and applications using Windows Portable Devices.
Requirements for Windows Media DRM-Enabled Applications	Properties that are required to enable Windows Media DRM-protected content transfers.
Samples	Description of two command-line desktop applications that are supplied with this software development kit.
Application Overview	Key concepts used in Windows Portable Devices.
Programming Guide	Key tasks that an application will perform, with step-by-step instructions and code snippets.
Programming Reference	Reference guide to the interfaces and data types defined by Windows Portable Devices.

Applications built on Windows Media Device Manager or Windows Image Acquisition can access Windows Portable Devices through a compatibility layer.

Microsoft provides several drivers for standard protocols and devices, including Media Transport Protocol (MTP) devices and Mass Storage Class (MSC) devices. If you are familiar with the User-Mode Driver Framework, you can develop your own drivers for custom devices.

Diagnostics

2/18/2021 • 3 minutes to read • Edit Online

Windows has APIs and services that support diagnostics in and of your desktop apps. They provide:

- Debugging and error handling.
- Support for profiling the performance of your apps.
- Support for troubleshooting and error reporting.
- System monitoring and event notification.
- Network monitoring and diagnostics.
- Assessment of system state.

In this section

TOPIC	DESCRIPTION
Assessment Execution Engine	The Windows Assessment Execution Engine (AXE) enables the management and execution of Windows system assessments. Assessments can help a person understand the state of a system and remedy problems with performance, reliability, or functionality. AXE provides infrastructure needed to manage assessments using a UX tool or script, run assessments, make measurements, process raw data into results, run diagnostics, and publish the results.
Debugging and Error Handling	Describes debugging and error handling.
Hardware Counter Profiling	Applications use the Hardware Counter Profiling (HCP) SDK to capture thread profiling data such as cycle time and the reasons for context switches. You can also use HCP to capture counter data for hardware performance counters that you have configured on the system.
Network Diagnostics Framework	The Network Diagnostics Framework (NDF) provides a way for component and application developers to simplify network troubleshooting for users. Users can attempt to diagnose and repair a network problem using a single troubleshooting tool.
Network Monitor	Network Monitor captures network traffic for display and analysis. It enables you to perform tasks such as analyzing previously captured data in user-defined methods and extract data from defined protocol parsers.
Performance Counters	Counters are used to provide information as to how well the operating system or an application, service, or driver is performing. The counter data can help determine system bottlenecks and fine-tune system and application performance. The operating system, network, and devices provide counter data that an application can consume to provide users with a graphical view of how well the system is performing.

TOPIC	DESCRIPTION
Performance Logs and Alerts	Performance Logs and Alerts (PLA) provides application programmers the ability to generate alert notifications based on performance counter thresholds. Programmers can also use PLA to query performance data, create event tracing sessions, capture a computer's configuration, and trace the API calls in some of the Win32 system DLLs.
Process Snapshotting	Process snapshotting enables you to capture process state, in part or whole. It is similar to the Tool Help API, but with one important advantage: it can efficiently capture the virtual address contents of a process using the Windows internal POSIX fork clone capability. The process snapshot can be dumped into a file using the **MiniDumpWriteDump** function.
Process Status API	The process status application programming interface (PSAPI) is a helper library that makes it easier for you to obtain information about processes and device drivers.
System Event Notification Service	Applications designed for use by mobile users require a unique set of connectivity functions and notifications. In the past these individual applications were required to implement these features internally. The System Event Notification Service (SENS) now provides these capabilities in the operating system, creating a uniform connectivity and notification interface for applications. Using SENS developers can determine connection bandwidth and latency information from within their application and optimize the application's operation based on those conditions.
System Monitor	System Monitor (SYSMON) is the application programming interface (API) that you use to configure the Microsoft System Monitor ActiveX control. The System Monitor control lets you view real-time and previously logged performance counter data such as memory, disk, and processor counter data.
Tool Help Library	The functions provided by the tool help library make it easier for you to obtain information about currently executing applications.
Windows Error Reporting	The error reporting feature enables users to notify Microsoft of application faults, kernel faults, unresponsive applications, and other application specific problems. Microsoft can use the error reporting feature to provide customers with troubleshooting information, solutions, or updates for their specific problems. Developers can use this infrastructure to receive information that can be used to improve their applications.
Windows Events	Describes event tracing and logging.
Windows Performance Analyzer (WPA)	Windows Performance Analyzer (WPA) is a set of performance monitoring tools used to produce in-depth performance profiles of Microsoft Windows operating systems and applications.

TOPIC	DESCRIPTION
Windows Performance Toolkit (WPT)	The Windows Performance Toolkit consists of performance monitoring tools that produce in-depth performance profiles of Microsoft Windows operating systems and applications. This documentation discusses both Windows Performance Recorder (WPR) and Windows Performance Analyzer (WPA).
Windows Troubleshooting Platform	Windows Troubleshooting Platform (WTP) provides ISVs, OEMs, and administrators the ability to write troubleshooting packs that are used to discover and resolve issues found on the computer.

Debugging and Error Handling

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Application Recovery and Restart
- Error Handling
- Basic Debugging
- Debug Help Library
- Structured Exception Handling
- Wait Chain Traversal
- Intel AVX

Application Recovery and Restart

2/22/2020 • 2 minutes to read • Edit Online

Purpose

An application can use Application Recovery and Restart (ARR) to save data and state information before the application exits due to an unhandled exception or when the application stops responding. The application is also restarted, if requested.

An application can also be restarted if an installer updates a component of the application, or if the computer needs to restart as the result of an update. Note that to support automatic application restart after an installer updates an application, both the application and installer need to be authored appropriately. For details, see Registering for Application Restart.

Developer audience

ARR is designed for C and C++ developers.

Run-time requirements

ARR is available starting with the Windows Vista operating system.

In this section

TOPIC	DESCRIPTION
Using Application Recovery and Restart	Procedural guide for registering for recovery and restart.
Application Recovery and Restart Reference	Reference information for the ARR API.

Error Handling (Error Handling)

2/18/2021 • 2 minutes to read • Edit Online

Well-written applications include error-handling code that allows them to recover gracefully from unexpected errors. When an error occurs, the application may need to request user intervention, or it may be able to recover on its own. In extreme cases, the application may log the user off or shut down the system.

- About Error Handling
- Using Error Handling
- Error Handling Reference

For information about exception handling, see Structured Exception Handling.

Network Diagnostics Framework

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The Network Diagnostics Framework (NDF) provides a way for component and application developers to simplify network troubleshooting for users. Users can attempt to diagnose and repair a network problem using a single troubleshooting tool.

Microsoft provides NDF helper classes, some of which are extensible so that developers can create troubleshooting units called helper class extensions to provide more detailed diagnoses specific to particular software or hardware components.

Where applicable

NDF can be used by any vendor's software which relies on network connectivity.

Developer audience

The NDF API is designed for C/C++ developers.

Run-time requirements

NDF is available for computers running Windows Vista, Windows Server 2008, or later.

In this section

TOPIC	DESCRIPTION
About NDF	Provides a general summary of the NDF technology.
Using NDF	Provides information and examples on using NDF functionality, as well as how to extend this functionality.
NDF Reference	Provides information about enumerations, functions, interfaces, and structures that support the use of NDF, as well as information about the extensible helper classes provided by Microsoft.
Network Tracing in Windows 7	Discusses network tracing and tools to use for troubleshooting.

Network Monitor

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Network Monitor captures network traffic for display and analysis. It enables you to perform tasks such as analyzing previously captured data in user-defined methods and extract data from defined protocol parsers.

Developer audience

All API sets provided by Network Monitor can be accessed using C/C++. Those developers also working with *network packet providers* must also be familiar with COM.

Run-time requirements

To call from the Network Monitor API, you must be running on Windows NT Server 4.0, Windows 2000 Server, or Windows Server 2003, or have Microsoft Systems Management Server installed.

The NPP driver and supported features include all versions of Windows NT 4.0 and Windows 2000 Server.

NOTE

Network Monitor 2.1 and earlier are not supported on Windows Vista and later.

In this section

TOPIC	DESCRIPTION
The Network Monitor SDK	Introduction to the Network Monitor SDK.
About Network Monitor 2.1	Network Monitor concepts and services.
Using Network Monitor 2.1	Task-related topics that describe how to use Network Monitor functions and COM components.
Reference	Reference topics for Network Monitor.
Glossary	Definitions and terminology for Network Monitor.

Performance Counters

2/18/2021 • 2 minutes to read • Edit Online

Windows Performance Counters provide a high-level abstraction layer that provides a consistent interface for collecting various kinds of system data such as CPU, memory, and disk usage. System administrators often use performance counters to monitor systems for performance or behavior problems. Software developers often use performance counters to examine the resource usage of their programs.

In this section

TOPIC	DESCRIPTION
About Performance Counters	General information about Performance Counters.
Using Performance Counters	Task-related topics that describe how to use Performance Counters.
Performance Counters Reference	Detailed descriptions of Performance Counters functions and other programming elements.

Process Status API

11/2/2020 • 2 minutes to read • Edit Online

The process status application programming interface (PSAPI) is a helper library that makes it easier for you to obtain information about processes and device drivers. For more information, see the following topics:

- About PSAPI
- Using PSAPI
- PSAPI Reference

These functions are available in Psapi.dll.

The same information is generally available through the performance data in the registry. For more information, see Performance Counters.

System Event Notification Service

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Applications designed for use by mobile users require a unique set of connectivity functions and notifications. In the past these individual applications were required to implement these features internally. The System Event Notification Service (SENS) now provides these capabilities in the operating system, creating a uniform connectivity and notification interface for applications. Using SENS developers can determine connection bandwidth and latency information from within their application and optimize the application's operation based on those conditions.

Where applicable

The connectivity functions and notifications of SENS are useful for applications written for mobile computers or computers connected to high latency local area networks.

Developer audience

This document is intended for software developers who develop applications for mobile computing and high latency local area networks. This document provides a complete reference of all parts of the System Event Notification Service including the SENS object and supporting methods.

Run-time requirements

Requires Microsoft Windows XP or later. For information about which operating systems are required to use a particular interface or function, see the Requirements section of the documentation.

In this section

TOPIC	DESCRIPTION
Overview	General information about System Event Notification Service.
Reference	Documentation of System Event Notification Service methods and interfaces.

Related topics

IEventSubscription

System Monitor

2/18/2021 • 2 minutes to read • Edit Online

Purpose

System Monitor (SYSMON) is the application programming interface (API) that you use to configure the Microsoft System Monitor ActiveX control. The System Monitor control lets you view real-time and previously logged performance counter data such as memory, disk, and processor counter data.

Where applicable

Use the SYSMON API when you add the System Monitor control to any ActiveX-enabled container such as a Microsoft Word document, or a web page viewed by Internet Explorer, and you want to modify its default behavior. For example, you can create a performance monitor that will only display a predefined set of system counters, or with predefined display settings.

Developer audience

SYSMON is designed for use by C, C++, Visual Basic, and scripting developers.

Run-time requirements

SYSMON is available starting with Microsoft Windows 2000.

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
Using System Monitor	Task-related topics that describe how to use the SYSMON API.
System Monitor Reference	Detailed descriptions of SYSMON interfaces and other programming elements.

Tool Help Library

2/22/2020 • 2 minutes to read • Edit Online

The functions provided by the tool help library make it easier for you to obtain information about currently executing applications. These functions are designed to streamline the creation of tools, specifically debuggers.

- About Tool Help Functions
- Using the Tool Help Functions
- Tool Help Reference

Windows Error Reporting

11/2/2020 • 2 minutes to read • Edit Online

The error reporting feature enables users to notify Microsoft of application faults, kernel faults, unresponsive applications, and other application specific problems. Microsoft can use the error reporting feature to provide customers with troubleshooting information, solutions, or updates for their specific problems. Developers can use this infrastructure to receive information that can be used to improve their applications.

Users can enable error reporting through the Windows user interface. They can choose to report errors for specific applications. Administrators can override these settings using Group Policy.

Developers can register with Windows Desktop Application Program to get information about the problems customers are experiencing with their applications and help customers fix these problems. Developers can also use Application Recovery and Restart to ensure that customers do not lose data when their application crashes and allow users to quickly return to their tasks.

In this Section

- What's New in WER
- About WER
- Using WER
- WER Reference

Related topics

Application Recovery and Restart

Windows Desktop Application Program

Windows Events

11/2/2020 • 2 minutes to read • Edit Online

Events are typically used for troubleshooting application and driver software.

- Prior to Windows Vista, you would use either Event Tracing for Windows (ETW) or Event Logging to log events.
- Windows Vista introduced a new event model that unified both the Event Tracing for Windows (ETW) and Windows Event Log API.
- Windows 10 introduces TraceLogging which builds on ETW and provides a simplified way to instrument code for native, .NET and WinRT developers.

The new TraceLogging model allows you to include structured data with events, correlate events, and does not require a separate instrumentation manifest XML file.

The Windows Vista model uses an XML manifest to define the events that you want to publish. Events can be published to a channel or an ETW session. You can publish the events to the following types of channels: Admin, Operational, Analytic and Debug. If you use only ETW to enable the publisher, you do not need to specify channels in your manifest. For complete details on writing a manifest, see Writing an Instrumentation Manifest, and for information on channels, see Defining Channels.

To register your event publisher and to publish events, you use the ETW API. For details, see Providing Events and Developing a Provider. The event publisher will automatically write the events to the channels specified in the manifest if they are enabled.

If you want to control the events that an event publisher publishes at a finer level of granularity, use the ETW API. For example, if the manifest defines both write and read events, you can enable only the write events. An event can also specify a level value such as warning or error, so you can limit the events that are written to those that specify the error level. For details, see Controlling Event Tracing Sessions. The events are written to the session's log file.

Consuming events involves retrieving the events from an event channel, an event log file (.evtx or .evt files), a trace file (.etl files), or a real-time ETW session. To consume events from an ETW trace file or a real-time ETW session, use the trace data helper (TDH) functions in ETW to consume the events. You can also use TDH to read the event metadata. For details, see Consuming Events. To consume events from an event channel or an event log file, use the Windows Event Log functions to query or subscribe to events. For more information, see Querying for Events or Subscribing to Events.

Prior to Windows Vista, you must use Event Tracing for Windows or Event Logging to publish and consume events.

TraceLogging

11/2/2020 • 2 minutes to read • Edit Online

Purpose

TraceLogging is the new Windows 10 event tracing framework for user-mode applications and kernel-mode drivers. TraceLogging builds on Event Tracing for Windows (ETW) and provides a simplified way to instrument code.

In this section

TOPIC	DESCRIPTION
About TraceLogging	TraceLogging is the new Windows 10 event tracing for user-mode applications and kernel-mode drivers. TraceLogging is a format for self-describing Event Tracing for Windows (ETW). TraceLogging builds on Event Tracing for Windows (ETW) and provides a simplified way to instrument code.
Using TraceLogging	The following topics provide a TraceLogging quick start for native and managed code, with examples.

TOPIC	DESCRIPTION
TraceLogging Reference	The following topics provide information about the native TraceLogging API. TraceLogging builds on Event Tracing for Windows (ETW) and provides a simplified way to instrument code. TraceLogging allows you to include structured data with events, correlate events, and does not require a separate instrumentation manifest XML file. For WinRT developers • **LoggingChannel** has been extended in Windows 10 to log self-describing Event Tracing for Windows (ETW) events without the need for a manifest. • **LoggingActivity** has been extended in Windows 10 to provide activity start and stop methods that provide control over the format and contents of the Start and Stop events. Additionally, activities can be nested. For managed code (Microsoft .NET Framework) developers • The EventSource class that shipped with previous versions of the .NET Framework already supports writing ETW events without the need for a manifest. However, developers were required to use EventSource as a base class and add attributes and methods to the derived class that were turned into an ETW manifest automatically. Now, developers do not have to derive from EventSource and can instead use EventSource directly to log self-describing events that do not require a manifest. For C/C++ developers • TraceLoggingProvider.h is the recommended API for C/C++ developers in user or kernel mode. This API is not well suited for use in dynamic (scripted) scenarios such as Javascript. The following links describe the C/C++ API.

Developer audience

TraceLogging is designed for use by user-mode application developers and kernel-mode driver developers who want to add tracing to their code.

Event Tracing

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Event Tracing for Windows (ETW) provides application programmers the ability to start and stop event tracing sessions, instrument an application to provide trace events, and consume trace events. Trace events contain an event header and provider-defined data that describes the current state of an application or operation. You can use the events to debug an application and perform capacity and performance analysis.

This documentation is for user-mode applications that want to use ETW. For information about instrumenting device drivers that run in kernel mode, see WPP Software Tracing and Adding Event Tracing to Kernel-Mode Drivers in the Windows Driver Kit (WDK).

Where applicable

Use ETW when you want to instrument your application, log user or kernel events to a log file, and consume events from a log file or in real time.

Developer audience

ETW is designed for C and C++ developers who write user-mode applications.

Run-time requirements

ETW is included in Microsoft Windows 2000 and later. For information about which operating systems are required to use a particular function, see the Requirements section of the documentation for the function.

Process ETW traces in .NET code

You can use the .NET TraceProcessing API to analyze ETW traces for your applications and other software components. This API is used internally at Microsoft to analyze ETW data produced the Windows engineering system, and it is also used to power several tables in Windows Performance Analyzer. This API is available as a NuGet package.

For more information, see this article.

In this section

TOPIC	DESCRIPTION
What's New in Event Tracing	New features that were added to Event Tracing in each release.
About Event Tracing	General information about Event Tracing.
Using Event Tracing	Task-related topics that describe how to use the ETW API.
Event Tracing Reference	Detailed descriptions of ETW functions and other programming elements.

Event Logging (Event Logging)

2/18/2021 • 2 minutes to read • Edit Online

Many applications record errors and events in proprietary error logs, each with their own format and user interface. Data from different applications can't easily be merged into one complete report, requiring system administrators or support representatives to check a variety of sources to diagnose problems.

Event logging provides a standard, centralized way for applications (and the operating system) to record important software and hardware events. The event logging service records events from various sources and stores them in a single collection called an *event log*. The Event Viewer enables you to view logs; the programming interface also enables you to examine logs.

- About Event Logging
- Using Event Logging
- Event Logging Reference

NOTE

The Event Logging API was designed for applications that run on the Windows Server 2003, Windows XP, or Windows 2000 operating system. In Windows Vista, the event logging infrastructure was redesigned. Applications that are designed to run on Windows Vista or later operating systems should use Windows Event Log to log events.

Windows Event Log

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Windows Event Log API defines the schema that you use to write an instrumentation manifest. An instrumentation manifest identifies your event provider and the events that it logs. The API also includes the functions that an event consumer, such as the Event Viewer, would use to read and render the events. To write the events defined in the manifest, use the functions included in the Event Tracing (ETW) API.

Windows Event Log supersedes the Event Logging API beginning with the Windows Vista operating system.

Developer audience

Windows Event Log is designed for C/C++ programmers.

Run-time requirements

Windows Event Log is included in the operating system beginning with Windows Vista and Windows Server 2008.

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

For complete version history, see What's New.

In this section

TOPIC	DESCRIPTION
Using Windows Event Log	Procedural guide that shows how to use the Windows Event Log API.
Windows Event Log Reference	The data types, functions, enumerations, structures, constants, and schemas that the API includes.

Windows Event Collector

11/2/2020 • 2 minutes to read • Edit Online

You can subscribe to receive and store events on a local computer (event collector) that are forwarded from a remote computer (event source). The Windows Event Collector functions support subscribing to events by using the WS-Management protocol. For more information about WS-Management, see About Windows Remote Management.

Event Forwarding and Event Collection Architecture

Event collection allows administrators to get events from remote computers and store them in a local event log on the collector computer. The destination log path for the events is a property of the subscription. All data in the forwarded event is saved in the collector computer event log (none of the information is lost). Additional information related to the event forwarding is also added to the event. For more information about how to enable a computer to receive collected events or forward events, see Configure Computers to Forward and Collect Events.

Subscriptions

The following list describes the types of event subscriptions:

- Source-initiated subscriptions: allows you to define an event subscription on an event collector computer without defining the event source computers. Multiple remote event source computers can then be set up (using a group policy setting) to forward events to the event collector computer. For more information, see Setting up a Source Initiated Subscription. This subscription type is useful when you do not know or you do not want to specify all the event sources computers that will forward events.
- Collector-initiated subscriptions: allows you to create an event subscription if you know all the event source computers that will forward events. You specify all the event sources at the time the subscription is created. For more information, see Creating a Collector Initiated Subscription.

Windows Event Collector Functions

For more information and code examples that use the Event Collector functions, see Using Windows Event Collector.

For more information about the functions used to collect and forward events, see Windows Event Collector functions.

Documents and Printing

2/18/2021 • 2 minutes to read • Edit Online

These topics describe the documents and printing features of Windows that enable applications to save, view, and print.

In this section

TOPIC	DESCRIPTION
What's New for Printing	Windows 8 supports printing for Windows Store apps using JavaScript and HTML and printing for Windows Store apps using C#/VB/C++ and XAML. Windows 8 also includes a new COM-based API that provides full support for Open XPS and access to portions of the new printer drivers that Windows 8 supports. For more information about the new API, see Print Document Package API. The Windows Print Driver Inbox Program makes sure that Windows 8 includes support for a high percentage of the popular printers.
XPS Documents	Information about the XPS Document API an XPS Digital Signatures. • XPS Document API The XPS Document API is a native Windows API that supports the XPS OM. The XPS Document API was introduced in Windows 7 and can be used in user-mode programs and XPSDrv printer drivers. • XPS Digital Signature API XPS Digital Signatures enable document signing, verification of the signer's identity, and indication of whether an XPS document has changed since it was signed.

TOPIC	DESCRIPTION
Printing	Information about the printing features supported by Windows. These features include: • Print Document Package API Provides apps with an interface that allows the management of the **PrintDocument** package. • Print Spooler API Provides an interface to the print spooler so that applications can manage printers and print jobs. • Print Ticket API Provides applications with functions to manage and convert print tickets. • GDI Print API Provides applications with a device-independent printing interface. • XPS Print API Provides an interface to the print spooler that applications can use to send XPS documents to a printer. [!Note] The XPS Print API is not supported and may be altered or unavailable in the future. Client applications should use the Print Document Package API instead.
Print Schema	The Print Schema and related technologies describe a printer's features and specify the printing preferences of a document to printers and viewing applications. The Print Schema Specification is a downloadable document that contains information about the Print Schema and how to use it in documents and printing. The online information that is found in this section is provided for your information only and might not accurately reflect the current version of the Print Schema Specification. Refer to the Print Schema Specification for the most current design information.

Additional resources

The Print Sample sample app demonstrates how to print from a Windows Store app starting with Windows 8.

The features described in this section are for native Windows programming. To use similar features in .NET (managed) programming, see Windows Presentation Foundation Documents.

XPS documents are built on the Packaging API. See the Packaging API, for lower level access to the contents of an XPS document.

Related topics

Bidirectional printer communications (Hardware Dev Center)

Graphics and gaming

11/2/2020 • 2 minutes to read • Edit Online

Windows provides APIs and components that support graphics, gaming, and imaging.

In this section

TOPIC	DESCRIPTION
DirectX Graphics and Gaming	DirectX graphics provides a set of APIs that you can use to create games and other high-performance multimedia applications.
DirectComposition	DirectComposition enables high-performance bitmap composition with transforms, effects, and animations. You can use the DirectComposition API to create visually engaging user interfaces that feature rich and fluid animated transitions from one visual to another.
Game Mode	The Game Mode APIs for the Universal Windows Platform (UWP) allow you to produce the most optimized gaming experience by taking advantage of Game Mode in Windows 10.
Gaming Device Information	The Gaming Device Information APIs allow UWP game developers to determine the type of console the game is running on, in order to make run-time choices on how to best use the hardware.
Windows Imaging Component (WIC)	The Windows Imaging Component (WIC) is an extensible platform that provides low-level API for digital images. WIC supports the standard web image formats, high dynamic range images, and raw camera data.
Win2D (External Site)	Win2D is an easy-to-use Windows Runtime API for immediate mode 2D graphics rendering with GPU acceleration. It is available to C# and C++ developers writing Windows apps for Windows 8.1, Windows Phone 8.1 and Windows 10. It utilizes the power of Direct2D, and integrates seamlessly with XAML and CoreWindow.
ANGLE for Windows Store (External Site)	ANGLE for Windows Store is an open-source project that allows developers to run OpenGL ES content on Windows by translating OpenGL ES API calls to DirectX 11 API calls. ANGLE for Windows Store supports Windows 8.1, Windows Phone 8.1, and Windows 10.

NOTE

As of Windows 10, version 1809 (10.0; Build 17763), TruePlay is removed from Windows. TruePlay documentation is not published.

Related topics

- Audio and Video

DirectX graphics and gaming

2/18/2021 • 2 minutes to read • Edit Online

This content focuses on using DirectX in a Win32 application. For information on using DirectX in a UWP application, see the Windows 10 game development guide (UWP).

In this section

TOPIC	DESCRIPTION
Getting Started with DirectX graphics	Microsoft DirectX graphics provides a set of APIs that you can use to create games and other high-performance multimedia apps. DirectX graphics includes support for high-performance 2-D and 3-D graphics.
Programming DirectX with COM	The Microsoft Component Object Model (COM) is an object-oriented programming model used by several technologies, including the bulk of the DirectX API surface.
Direct2D	Direct2D is a hardware-accelerated, immediate-mode, 2-D graphics API that provides high performance and high-quality rendering for 2-D geometry, bitmaps, and text.
Direct3D	Direct3D enables you to create 3-D graphics for games and scientific apps.
DXCore	DXCore is an adapter enumeration API for graphics and compute devices, so some of its facilities overlap with those of DXGI.
DirectWrite	DirectWrite supports high-quality text rendering, resolution-independent outline fonts, and full Unicode text and layouts.
DirectXMath	DirectXMath provides an optimal and portable interface for arithmetic and linear algebra operations on single-precision floating-point vectors (2D, 3D, and 4D) or matrices (3×3 and 4×4).
windowsnumerics.h APIs	The `windowsnumerics.h` header file defines C++ vector and matrix types in the Windows.Foundation.Numerics namespace.
Classic DirectX Graphics	Microsoft DirectX graphics technologies that are currently minimally used. We do not recommend using these classic DirectX graphics technologies for new apps.
Tools for DirectX Graphics	Describes tools for DirectX Graphics.
DirectX Graphics Articles	Contains technical articles for DirectX Graphics.
XAudio2 APIs	Provides a signal processing and mixing foundation for games. XAudio2 replaces DirectSound.

TOPIC	DESCRIPTION
XInput game controller APIs	Describes how to use the XInput API to interact with the Xbox 360 Controller when it is connected to a Windows computer. XInput replaces DirectInput.

Related topics

- Audio and Video
- Graphics and Gaming

Getting started with DirectX Graphics

2/18/2021 • 2 minutes to read • Edit Online

Microsoft DirectX graphics provides a set of APIs that you can use to create games and other high-performance multimedia applications. DirectX graphics includes support for high-performance 2-D and 3-D graphics.

For 3-D graphics, use the Microsoft Direct3D 11 API. Even if you have Microsoft Direct3D 9-level or Microsoft Direct3D 10-level hardware, you can use the Direct3D 11 API and target a feature level 9_x or feature level 10_x device. For info about how to develop 3-D graphics with DirectX, see Create 3D graphics with DirectX.

For 2-D graphics and text, use Direct2D and DirectWrite rather than Windows Graphics Device Interface (GDI).

To compose bitmaps that Direct3D 11 or Direct2D populated, use DirectComposition.

To learn about how to create a Windows Store app that uses DirectX, see Create your first Windows Store app using DirectX. You can use the **Windows.UI::Xaml::Controls::SwapChainPanel** class to create high-performance DirectX apps with a XAML UI overlay. For more info about combining XAML and DirectX in a Windows app, see DirectX and XAML interop.

To learn about how to build a display driver for Windows 8, see Roadmap for the Windows Display Driver Model (WDDM).

If you need the documentation for previous DirectX versions, see Classic DirectX Graphics.

Programming DirectX with COM

2/18/2021 • 23 minutes to read • Edit Online

The Microsoft Component Object Model (COM) is an object-oriented programming model used by several technologies, including the bulk of the DirectX API surface. For that reason, you (as a DirectX developer) inevitably use COM when you program DirectX.

NOTE

The topic Consume COM components with C++/WinRT shows how to consume DirectX APIs (and any COM API, for that matter) by using C++/WinRT. That's by far the most convenient and recommended technology to use.

Alternatively, you can use raw COM, and that's what this topic is about. You'll need a basic understanding of the principles and programming techniques involved in consuming COM APIs. Although COM has a reputation for being difficult and complex, the COM programming required by most DirectX applications is straightforward. In part, this is because you'll be consuming the COM objects provided by DirectX. There's no need for you to author your own COM objects, which is typically where the complexity arises.

COM component overview

A COM object is essentially an encapsulated component of functionality that can be used by applications to perform one or more tasks. For deployment, one or more COM components are packaged into a binary called a COM server; more often than not a DLL.

A traditional DLL exports free functions. A COM server can do the same. But the COM components inside the COM server expose COM interfaces and member methods belonging to those interfaces. Your application creates instances of COM components, retrieves interfaces from them, and calls methods on those interfaces in order to benefit from the features implemented in the COM components.

In practice, this feels similar to calling methods on a regular C++ object. But there are some differences.

- A COM object enforces stricter encapsulation than a C++ object does. You can't just create the object, and then call any public method. Instead, a COM component's public methods are grouped into one or more COM interfaces. To call a method, you create the object and retrieve from the object the interface that implements the method. An interface typically implements a related set of methods that provide access to a particular feature of the object. For example, the **ID3D12Device** interface represents a virtual graphics adapter, and it contains methods that enable you create resources, for example, and many other adapter-related tasks.
- A COM object is not created in the same way as a C++ object. There are several ways to create a COM object, but all involve COM-specific techniques. The DirectX API includes a variety of helper functions and methods that simplify creating most of the DirectX COM objects.
- You must use COM-specific techniques to control the lifetime of a COM object.
- The COM server (typically a DLL) doesn't need to be explicitly loaded. Nor do you link to a static library in order to use a COM component. Each COM component has a unique registered identifier (a globally-unique identifier, or GUID), which your application uses to identify the COM object. Your application identifies the component, and the COM runtime automatically loads the correct COM server DLL.
- COM is a binary specification. COM objects can be written in and accessed from a variety of languages. You don't need to know anything about the object's source code. For example, Visual Basic applications routinely use COM objects that were written in C++.

Component, object, and interface

It's important to understand the distinction between components, objects, and interfaces. In casual usage, you may hear a component or object referred to by the name of its principal interface. But the terms are not interchangeable. A component can implement any number of interfaces; and an object is an instance of a component. For example, while all components must implement the **IUnknown interface**, they normally implement at least one additional interface, and they might implement many.

To use a particular interface method, you must not only instantiate an object, you must also obtain the correct interface from it.

In addition, more than one component might implement the same interface. An interface is a group of methods that perform a logically-related set of operations. The interface definition specifies only the syntax of the methods and their general functionality. Any COM component that needs to support a particular set of operations can do so by implementing a suitable interface. Some interfaces are highly specialized, and are implemented only by a single component; others are useful in a variety of circumstances, and are implemented by many components.

If a component implements an interface, it must support every method in the interface definition. In other words, you must be able to call any method and be confident that it exists. However, the details of how a particular method is implemented may vary from one component to another. For example, different components may use different algorithms to arrive at the final result. There is also no guarantee that a method will be supported in a nontrivial way. Sometimes, a component implements a commonly-used interface, but it needs to support only a subset of the methods. You will still be able to call the remaining methods successfully, but they will return an **HRESULT** (which is a standard COM type representing a result code) containing the value **E_NOTIMPL**. You should refer to its documentation to see how an interface is implemented by any particular component.

The COM standard requires that an interface definition must not change once it has been published. The author cannot, for example, add a new method to an existing interface. The author must instead create a new interface. While there are no restrictions on what methods must be in that interface, a common practice is to have the next-generation interface include all the of the old interface's methods, plus any new methods.

It's not unusual for an interface to have several generations. Typically, all generations perform essentially the same overall task, but they're different in specifics. Often, a COM component implements every current and prior generation of a given interface's lineage. Doing so allows older applications to continue using the object's older interfaces, while newer applications can take advantage of the features of the newer interfaces. Typically, a descent group of interfaces all have the same name, plus an integer that indicates the generation. For example, if the original interface were named **IMyInterface** (implying generation 1), then the next two generations would be called **IMyInterface2** and **IMyInterface3**. In the case of DirectX interfaces, successive generations are typically named for the version number of DirectX.

GUIDs

GUIDs are a key part of the COM programming model. At its most basic, a GUID is a 128-bit structure. However, GUIDs are created in such a way as to guarantee that no two GUIDs are the same. COM uses GUIDs extensively for two primary purposes.

- To uniquely identify a particular COM component. A GUID that is assigned to identify a COM component is called a class identifier (CLSID), and you use a CLSID when you want to create an instance of the associated COM component.
- To uniquely identify a particular COM interface. A GUID that is assigned to identify a COM interface is called an interface identifier (IID), and you use an IID when you request a particular interface from an instance of a component (an object). An interface's IID will be the same, regardless of which component implements the interface.

For convenience, the DirectX documentation normally refers to components and interfaces by their descriptive names (for example, **ID3D12Device**) rather than by their GUIDs. Within the context of the DirectX documentation, there is no ambiguity. It's technically possible for a third-party to author an interface with the descriptive name **ID3D12Device** (it would need to have a different IID in order to be valid). In the interest of clarity, though, we don't recommend that.

So, the only unambiguous way to refer to a particular object or interface is by its GUID.

Although a GUID is a structure, a GUID is often expressed in equivalent string form. The general format of the string form of a GUID is 32 hexadecimal digits, in the format 8-4-4-4-12. That is, {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}, where each x corresponds to a hexadecimal digit. For example, the string form of the IID for the **ID3D12Device** interface is {189819F1-1DB6-4B57-BE54-1821339B85F7}.

Because the actual GUID is somewhat clumsy to use and easy to mistype, an equivalent name is usually provided as well. In your code, you can use this name instead of the actual structure when you call functions, for example when you pass an argument for the `riid` parameter to **D3D12CreateDevice**. The customary naming convention is to prepend either IID_ or CLSID_ to the descriptive name of the interface or object, respectively. For example, the name of the **ID3D12Device** interface's IID is IID_ID3D12Device.

NOTE

DirectX applications should link with `dxguid.lib` and `uuid.lib` to provide definitions for the various interface and class GUIDs. Visual C++ and other compilers support the **__uuidof** operator language extension, but explicit C-style linkage with these link libraries is also supported and fully portable.

HRESULT values

Most COM methods return a 32-bit integer called an **HRESULT**. With most methods, the HRESULT is essentially a structure that contains two primary pieces of information.

- Whether the method succeeded or failed.
- More detailed information about the outcome of the operation performed by the method.

Some methods return a **HRESULT** value from the standard set defined in `Winerror.h`. However, a method is free to return a custom **HRESULT** value with more specialized information. These values are normally documented on the method's reference page.

The list of **HRESULT** values that you find on a method's reference page is often only a subset of the possible values that may be returned. The list typically covers only those values that are specific to the method, as well as those standard values that have some method-specific meaning. You should assume that a method may return a variety of standard **HRESULT** values, even if they're not explicitly documented.

While **HRESULT** values are often used to return error information, you should not think of them as error codes. The fact that the bit that indicates success or failure is stored separately from the bits that contain the detailed information allows **HRESULT** values to have any number of success and failure codes. By convention, the names of success codes are prefixed by S_ and failure codes by E_. For example, the two most commonly used codes are S_OK and E_FAIL, which indicate simple success or failure, respectively.

The fact that COM methods may return a variety of success or failure codes means that you have to be careful how you test the **HRESULT** value. For example, consider a hypothetical method with documented return values of S_OK if successful and E_FAIL if not. However, remember that the method may also return other failure or success codes. The following code fragment illustrates the danger of using a simple test, where `hr` contains the **HRESULT** value returned by the method.

```
if (hr == E_FAIL)
{
    // Handle the failure case.
}
else
{
    // Handle the success case.
}
```

As long as, in the failure case, this method only ever return E_FAIL (and not some other failure code), then this test works. However, it's more realistic that a given method is implemented to return a set of specific failure codes, perhaps E_NOTIMPL or E_INVALIDARG. With the code above, those values would be incorrectly interpreted as a success.

If you need detailed information about the outcome of the method call, you need to test each relevant **HRESULT** value. However, you may be interested only in whether the method succeeded or failed. A robust way to test whether an **HRESULT** value indicates success or failure is to pass the value to the one of the following macros, defined in Winerror.h.

- The `SUCCEEDED` macro returns TRUE for a success code, and FALSE for a failure code.
- The `FAILED` macro returns TRUE for a failure code, and FALSE for a success code.

So, you can fix the preceding code fragment by using the `FAILED` macro, as shown in the following code.

```
if (FAILED(hr))
{
    // Handle the failure case.
}
else
{
    // Handle the success case.
}
```

This corrected code fragment properly treats E_NOTIMPL and E_INVALIDARG as failures.

Although most COM methods return structured **HRESULT** values, a small number use the **HRESULT** to return a simple integer. Implicitly, these methods are always successful. If you pass an **HRESULT** of this sort to the SUCCEEDED macro, then the macro always returns TRUE. An example of a commonly-called method that doesn't return an **HRESULT** is the **IUnknown::Release** method, which returns a ULONG. This method decrements an object's reference count by one and returns the current reference count. See Managing a COM object's lifetime for a discussion of reference counting.

The address of a pointer

If you view a few COM method reference pages, you'll probably run across something like the following.

```
HRESULT D3D12CreateDevice(
  IUnknown          *pAdapter,
  D3D_FEATURE_LEVEL MinimumFeatureLevel,
  REFIID            riid,
  void              **ppDevice
);
```

While a normal pointer is quite familiar to any C/C++ developer, COM often uses an additional level of indirection. This second level of indirection is indicated by two asterisks, `**`, following the type declaration, and the variable name typically has a prefix of `pp`. For the function above, the `ppDevice` parameter is typically referred to as the address of a pointer to a void. In practice, in this example, `ppDevice` is the address of a pointer to an **ID3D12Device** interface.

Unlike a C++ object, you don't access a COM object's methods directly. Instead, you must obtain a pointer to an interface that exposes the method. To invoke the method, you use essentially the same syntax as you would to invoke a pointer to a C++ method. For example, to invoke the **IMyInterface::DoSomething** method, you would use the following syntax.

```
IMyInterface * pMyIface = nullptr;
...
pMyIface->DoSomething(...);
```

The need for a second level of indirection comes from the fact that you don't create interface pointers directly. You must call one of a variety of methods, such as the **D3D12CreateDevice** method shown above. To use such a method to obtain an interface pointer, you declare a variable as a pointer to the desired interface, and then you pass the address of that variable to the method. In other words, you pass the address of a pointer to the method. When the method returns, the variable points to the requested interface, and you can use that pointer to call any of the interface's methods.

```
IDXGIAdapter * pIDXGIAdapter = nullptr;
...
ID3D12Device * pD3D12Device = nullptr;
HRESULT hr = ::D3D12CreateDevice(
    pIDXGIAdapter,
    D3D_FEATURE_LEVEL_11_0,
    IID_ID3D12Device,
    &pD3D12Device);
if (FAILED(hr)) return E_FAIL;

// Now use pD3D12Device in the form pD3D12Device->MethodName(...);
```

Creating a COM object

There are several ways to create a COM object. These are the two most commonly used in DirectX programming.

- Indirectly, by calling a DirectX method or function that creates the object for you. The method creates the object, and returns an interface on the object. When you create an object this way, sometimes you can specify which interface should be returned, other times the interface is implied. The code example above shows how to indirectly create a Direct3D 12 device COM object.
- Directly, by passing the object's CLSID to the CoCreateInstance function. The function creates an instance of the object, and it returns a pointer to an interface that you specify.

One time, before you create any COM objects, you must initialize COM by calling the CoInitializeEx function. If you're creating objects indirectly, then the object creation method handles this task. But, if you need to create an object with **CoCreateInstance**, then you must call **CoInitializeEx** explicitly. When you're finished, COM must be uninitialized by calling CoUninitialize. If you make a call to **CoInitializeEx** then you must match it with a call to **CoUninitialize**. Typically, applications that need to explicitly initialize COM do so in their startup routine, and they uninitialize COM in their cleanup routine.

To create a new instance of a COM object with **CoCreateInstance**, you must have the object's CLSID. If this CLSID is publicly available, you will find it in the reference documentation or the appropriate header file. If the CLSID is not publicly available, then you can't create the object directly.

The **CoCreateInstance** function has five parameters. For the COM objects you will be using with DirectX, you can normally set the parameters as follows.

rclsid Set this to the CLSID of the object that you want to create.

pUnkOuter Set to `nullptr`. This parameter is used only if you are aggregating objects. A discussion of COM aggregation is outside the scope of this topic.

dwClsContext Set to CLSCTX_INPROC_SERVER. This setting indicates that the object is implemented as a DLL and runs as part of your application's process.

riid Set to the IID of the interface that you would like to have returned. The function will create the object and return the requested interface pointer in the ppv parameter.

ppv Set this to the address of a pointer that will be set to the interface specified by `riid` when the function returns. This variable should be declared as a pointer to the requested interface, and the reference to the pointer in the parameter list should be cast as (LPVOID *).

Creating an object indirectly is usually much simpler, as we saw in the code example above. You pass the object creation method the address of an interface pointer, and the method then creates the object and returns an interface pointer. When you create an object indirectly, even if you can't choose which interface the method returns, often you can still specify a variety of things about how the object should be created.

For example, you can pass to **D3D12CreateDevice** a value specifying the minimum D3D feature level that the returned device should support, as shown in the code example above.

Using COM interfaces

When you create a COM object, the creation method returns an interface pointer. You can then use that pointer to access any of the interface's methods. The syntax is identical to that used with a pointer to a C++ method.

Requesting Additional Interfaces

In many cases, the interface pointer that you receive from the creation method may be the only one that you need. In fact, it's relatively common for an object to export only one interface other than **IUnknown**. However, many objects export multiple interfaces, and you may need pointers to several of them. If you need more interfaces than the one returned by the creation method, there's no need to create a new object. Instead, request another interface pointer by using the object's **IUnknown::QueryInterface method**.

If you create your object with **CoCreateInstance**, then you can request an **IUnknown** interface pointer and then call **IUnknown::QueryInterface** to request every interface you need. However, this approach is inconvenient if you need only a single interface, and it doesn't work at all if you use an object creation method that doesn't allow you to specify which interface pointer should be returned. In practice, you usually don't need to obtain an explicit **IUnknown** pointer, because all COM interfaces extend the **IUnknown** interface.

Extending an interface is conceptually similar to inheriting from a C++ class. The child interface exposes all of the parent interface's methods, plus one or more of its own. In fact, you will often see "inherits from" used instead of "extends". What you need to remember is that the inheritance is internal to the object. Your application can't inherit from or extend an object's interface. However, you can use the child interface to call any of the methods of the child or parent.

Because all interfaces are children of **IUnknown**, you can call **QueryInterface** on any of the interface pointers that you already have for the object. When you do so, you must provide the IID of the interface that you're requesting and the address of a pointer that will contain the interface pointer when the method returns.

For example, the following code fragment calls **IDXGIFactory2::CreateSwapChainForHwnd** to create a primary swap chain object. This object exposes several interfaces. The **CreateSwapChainForHwnd** method returns an **IDXGISwapChain1** interface. The subsequent code then uses the **IDXGISwapChain1** interface to call **QueryInterface** to request an **IDXGISwapChain3** interface.

```
HRESULT hr = S_OK;

IDXGISwapChain1 * pDXGISwapChain1 = nullptr;
hr = pIDXGIFactory->CreateSwapChainForHwnd(
    pCommandQueue, // For D3D12, this is a pointer to a direct command queue.
    hWnd,
    &swapChainDesc,
    nullptr,
    nullptr,
    &pDXGISwapChain1));
if (FAILED(hr)) return hr;

IDXGISwapChain3 * pDXGISwapChain3 = nullptr;
hr = pDXGISwapChain1->QueryInterface(IID_IDXGISwapChain3, (LPVOID*)&pDXGISwapChain3);
if (FAILED(hr)) return hr;
```

NOTE

In C++ you can make use of the `IID_PPV_ARGS` macro rather than the explicit IID and cast pointer: `pDXGISwapChain1->QueryInterface(IID_PPV_ARGS(&pDXGISwapChain3));`. This is often used for creation methods as well as **QueryInterface**. See combaseapi.h for more information.

Managing a COM object's lifetime

When an object is created, the system allocates the necessary memory resources. When an object is no longer needed, it should be destroyed. The system can use that memory for other purposes. With C++ objects, you can control the object's lifetime directly with the `new` and `delete` operators in cases where you're operating at that level, or just by using the stack and scope lifetime. COM doesn't enable you to directly create or destroy objects. The reason for this design is that the same object may be used by more than one part of your application or, in some cases, by more than one application. If one of those references were to destroy the object, then the other references would become invalid. Instead, COM uses a system of reference counting to control an object's lifetime.

An object's reference count is the number of times one of its interfaces has been requested. Each time that an interface is requested, the reference count is incremented. An application releases an interface when that interface is no longer needed, decrementing the reference count. As long as the reference count is greater than zero, the object remains in memory. When the reference count reaches zero, the object destroys itself. You don't need to know anything about the reference count of an object. As long as you obtain and release an object's interfaces properly, the object will have the appropriate lifetime.

Properly handling reference counting is a crucial part of COM programming. Failure to do so can easily create a memory leak or a crash. One of the most common mistakes that COM programmers make is failing to release an interface. When this happens, the reference count never reaches zero, and the object remains in memory indefinitely.

NOTE

Direct3D 10 or later has slightly modified lifetime rules for objects. In particular, objects that are derived from **ID3DxxDeviceChild** never outlive their parent device (that is, if the owning **ID3DxxDevice** hits a 0 refcount, then all child objects are immediately invalid as well). Also, when you use **Set** methods to bind objects to the render pipeline, these references don't increase the reference count (that is, they are weak references). In practice, this is best handled by ensuring that you release all device child objects fully before you release the device.

Incrementing and decrementing the reference count

Whenever you obtain a new interface pointer, the reference count must be incremented by a call to IUnknown::AddRef. However, your application doesn't usually need to call this method. If you obtain an interface pointer by calling an object creation method, or by calling **IUnknown::QueryInterface**, then the object automatically increments the reference count. However, if you create an interface pointer in some other way, such as copying an existing pointer, then you must explicitly call **IUnknown::AddRef**. Otherwise, when you release the original interface pointer, the object may be destroyed even though you may still need to use the copy of the pointer.

You must release all interface pointers, regardless of whether you or the object incremented the reference count. When you no longer need an interface pointer, call IUnknown::Release to decrement the reference count. A common practice is to initialize all interface pointers to `nullptr`, and then to set them back to `nullptr` when they are released. That convention allows you to test all interface pointers in your cleanup code. Those that are not `nullptr` are still active, and you need to release them before you terminate the application.

The following code fragment extends the sample shown earlier to illustrate how to handle reference counting.

```
HRESULT hr = S_OK;

IDXGISwapChain1 * pDXGISwapChain1 = nullptr;
hr = pIDXGIFactory->CreateSwapChainForHwnd(
    pCommandQueue, // For D3D12, this is a pointer to a direct command queue.
    hWnd,
    &swapChainDesc,
    nullptr,
    nullptr,
    &pDXGISwapChain1));
if (FAILED(hr)) return hr;

IDXGISwapChain3 * pDXGISwapChain3 = nullptr;
hr = pDXGISwapChain1->QueryInterface(IID_IDXGISwapChain3, (LPVOID*)&pDXGISwapChain3);
if (FAILED(hr)) return hr;

IDXGISwapChain3 * pDXGISwapChain3Copy = nullptr;

// Make a copy of the IDXGISwapChain3 interface pointer.
// Call AddRef to increment the reference count and to ensure that
// the object is not destroyed prematurely.
pDXGISwapChain3Copy = pDXGISwapChain3;
pDXGISwapChain3Copy->AddRef();
...
// Cleanup code. Check to see whether the pointers are still active.
// If they are, then call Release to release the interface.
if (pDXGISwapChain1 != nullptr)
{
    pDXGISwapChain1->Release();
    pDXGISwapChain1 = nullptr;
}
if (pDXGISwapChain3 != nullptr)
{
    pDXGISwapChain3->Release();
    pDXGISwapChain3 = nullptr;
}
if (pDXGISwapChain3Copy != nullptr)
{
    pDXGISwapChain3Copy->Release();
    pDXGISwapChain3Copy = nullptr;
}
```

COM Smart Pointers

The code so far has explicitly called `Release` and `AddRef` to maintain the reference counts using **IUnknown** methods. This pattern requires the programmer to be diligent in remembering to properly maintain the count in all possible codepaths. This can result in complicated error-handling, and with C++ exception handling enabled can be particularly difficult to implement. A better solution with C++ is to make use of a smart pointer.

- **winrt::com_ptr** is a smart pointer provided by the C++/WinRT language projections. This is the recommended COM smart pointer to use for UWP apps. Note that C++/WinRT requires C++17.
- **Microsoft::WRL::ComPtr** is a smart pointer provided by the Windows Runtime C++ Template Library (WRL). This library is "pure" C++ so it can be utilized for Windows Runtime applications (via C++/CX or C++/WinRT) as well as classic Win32 desktop applications. This smart pointer also works on older versions of Windows that do not support the Windows Runtime APIs. For Win32 desktop applications, you can use `#include <wrl/client.h>` to only include this class and optionally define the preprocessor symbol `__WRL_CLASSIC_COM_STRICT__` as well. For more information, see COM smart pointers revisited.
- **CComPtr** is a smart pointer provided by the Active Template Library (ATL). The **Microsoft::WRL::ComPtr** is a newer version of this implementation that addresses a number of subtle usage issues, so use of this smart pointer is not recommended for new projects. For more information, see How to create and use CComPtr and CComQIPtr.

Using ATL with DirectX 9

To use the Active Template Library (ATL) with DirectX 9, you must redefine the interfaces for ATL compatibility. This allows you to properly use the **CComQIPtr** class to obtain a pointer to an interface.

You'll know if you don't redefine the interfaces for ATL, because you'll see the following error message.

```
[...]\atlmfc\include\atlbase.h(4704) :   error C2787: 'IDirectXFileData' : no GUID has been associated with
this object
```

The following code sample shows how to define the IDirectXFileData interface.

```
// Explicit declaration
struct __declspec(uuid("{3D82AB44-62DA-11CF-AB39-0020AF71E433}")) IDirectXFileData;

// Macro method
#define RT_IID(iid_, name_) struct __declspec(uuid(iid_)) name_
RT_IID("{1DD9E8DA-1C77-4D40-B0CF-98FEFDFF9512}", IDirectXFileData);
```

After redefining the interface, you must use the **Attach** method to attach the interface to the interface pointer returned by **::Direct3DCreate9**. If you don't, then the **IDirect3D9** interface won't be properly released by the smart pointer class.

The **CComPtr** class internally calls **IUnknown::AddRef** on the interface pointer when the object is created and when an interface is assigned to the **CComPtr** class. To avoid leaking the interface pointer, don't call **IUnknown::AddRef on the interface returned from **::Direct3DCreate9**.

The following code properly releases the interface without calling **IUnknown::AddRef**.

```
CComPtr<IDirect3D9> d3d;
d3d.Attach(::Direct3DCreate9(D3D_SDK_VERSION));
```

Use the previous code. Don't use the following code, which calls **IUnknown::AddRef** followed by **IUnknown::Release**, and doesn't release the reference added by **::Direct3DCreate9**.

```
CComPtr<IDirect3D9> d3d = ::Direct3DCreate9(D3D_SDK_VERSION);
```

Note that this is the only place in Direct3D 9 where you'll have to use the **Attach** method in this manner.

For more information about the **CComPTR** and **CComQIPtr** classes, see their definitions in the `Atlbase.h` header file.

Direct2D

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Direct2D is a hardware-accelerated, immediate-mode, 2-D graphics API that provides high performance and high-quality rendering for 2-D geometry, bitmaps, and text. The Direct2D API is designed to interoperate well with GDI, GDI+, and Direct3D.

Developer audience

Direct2D is designed primarily for use by the following classes of developers:

- Developers of large, enterprise-scale, native applications.
- Developers who create control toolkits and libraries for consumption by downstream developers.
- Developers who require server-side rendering of 2-D graphics.
- Developers who use Direct3D graphics and need simple, high-performance 2-D and text rendering for menus, user-interface (UI) elements, and Heads-up Displays (HUDs).

Run-time requirements

- Windows 7 or Windows Vista with Service Pack 2 (SP2) and Platform Update for Windows Vista and later.
- Windows Server 2008 R2 or Windows Server 2008 with Service Pack 2 (SP2) and Platform Update for Windows Server 2008 and later.

NOTE

The Platform Update for Windows Vista and Platform Update for Windows Server 2008 are a set of run-time libraries that enables developers to target applications to Windows 7, Windows Vista, Windows Server 2008 R2, and Windows Server 2008. These updates will be available to all Windows Vista and Windows Server 2008 customers through Windows Update. Third-party applications that require Platform Update for Windows Vista or Platform Update for Windows Server 2008 can have Windows Update detect whether the required updated is installed; if it is not, Windows Update will download and install it in the background. For more information about both updates, see Platform Update for Windows Vista.

In this section

TOPIC	DESCRIPTION
What's new in Direct2D	Here are some of the new additions to Direct2D.
About Direct2D	Introduces Direct2D, an API that provides Win32 developers with the ability to perform 2-D graphics rendering tasks with superior performance and visual quality.
Direct2D Quickstart for Windows 8	Summarizes the steps required to draw with Direct2D and provides example code.

TOPIC	DESCRIPTION
Getting Started with Direct2D	Describes how to get started creating Direct2D applications and provides example code.
Programming Guide	This section contains conceptual programming topics that describe how to use the Direct2D API.
Direct2D reference	Describes the Direct2D API in detail.
Tools and Utilities	Tools and utilities provided for Direct2D.
Samples	Sample applications that demonstrate the Direct2D API.
Direct2D Glossary	Describes terms commonly used by the Direct2D documentation.

Additional resources

- DirectX Graphics and Gaming
- DirectWrite

Direct3D

2/18/2021 • 2 minutes to read • Edit Online

Direct3D is a low-level API for drawing primitives with the rendering pipeline, or for performing parallel operations with the compute shader. See the content below for more information.

For info about obtaining and installing Direct3D, see Direct3D 12 programming environment setup.

In this section

TOPIC	DESCRIPTION
Getting started with Direct3D	Discusses Direct3D in more depth, different application models, different versions, rendering, and compute.
Direct3D 12 graphics	Direct3D 12 provides an API and platform that allows your application to take advantage of the graphics and computing capabilities of PCs equipped with one or more Direct3D 12-compatible GPUs.
Direct3D 11 Graphics	You can use Microsoft Direct3D 11 graphics to create 3-D graphics for games and scientific and desktop applications.
DXGI	DXGI handles enumerating graphics adapters, enumerating display modes, selecting buffer formats, sharing resources between processes, and presenting rendered frames to a window or monitor for display.
HLSL	HLSL is the high-level shader language for DirectX. Using HLSL, you can create C-like programmable shaders for the Direct3D pipeline.
DDS	The DirectDraw surface file format (DDS) supports uncompressed and compressed (DXTn) textures, mipmaps, cube maps, and volume maps. It's supported by DirectXTex, DirectXTK, legacy D3DX, and other DirectX tools.

Getting started with Direct3D

2/18/2021 • 2 minutes to read • Edit Online

Direct3D is a low-level API for drawing primitives with the rendering pipeline, or for performing parallel operations with the compute shader.

What is Direct3D?

Direct3D is a low-level API that you can use to draw triangles, lines, or points per frame, or to start highly parallel operations on the GPU.

Direct3D:

- Hides different GPU implementations behind a coherent abstraction. But you still need to know how to draw 3D graphics.
- Is designed to drive a separate graphics-specific processor. Newer GPUs have hundreds or thousands of parallel processors.
- Emphasizes parallel processing. You set up a bunch of rendering or compute state and then start an operation. You don't wait for immediate feedback from the operation. You don't mix CPU and GPU operations.

Which Direct3D APIs can you use?

The Direct3D APIs that you choose depend on the style of app you want to write.

- If you want to write a UWP app, use a subset of Direct3D 11, DXGI, and HLSL APIs. For a list of these APIs, see Win32 and COM APIs for UWP apps. To learn how to write a Direct3D 11 Windows Store app, see Create 3D graphics with DirectX.
- If you write a desktop app, you can use the full set of Direct3D 11, DXGI, and HLSL APIs.
- Starting with Windows 8, we no longer actively support the XNA framework for desktop apps. But Windows Store apps, UWP apps, and desktop apps can use the full set of the XAudio2 and DirectXMath APIs. Desktop apps can use the full set of the XInput APIs, while Windows Store apps and UWP apps can use most of the XInput APIs; for more info, see XInput Versions.

Which Direct3D version?

The Direct3D API version that you choose depends on the operating system and hardware level that you want to target.

- If you want to target Windows 8 and later, use Direct3D 11 APIs.
- Use Direct3D 9 APIs with Windows XP and later. All hardware supports Direct3D 9 APIs, even newer Direct3D 11-level hardware.
- Use Direct3D 10 APIs with Windows Vista and later. Only Direct3D 10-level and later hardware support Direct3D 10 APIs.
- Use Direct3D 10.1 and Direct3D 11 APIs with Windows 7 and later. You can also use Direct3D 10.1 and Direct3D 11 APIs with Windows Vista with Service Pack 2 (SP2) by downloading KB 971644.

Direct3D Rendering Pipeline

In the Direct3D rendering pipeline, data flows from several sources, like the tributaries of a river.

- Some parts of the flow are programmable.
- Some parts have knobs and dials.
- Sources of data are either serial streams of packets (vertices) or indexable arrays (shader resources).
- Vertices and shader resources flow into primitives, which you can amplify.
- Primitives and shader resources flow into pixel operations.

Direct3D Compute Shader

With the Direct3D compute shader, all the GPU's processors execute in parallel. So the compute shader behaves more like a pond than a river.

Direct3D 12 graphics

2/22/2020 • 2 minutes to read • Edit Online

This programming guide contains information about how to use the Direct3D 12 programmable pipeline to create a customized graphics engine.

The Direct3D 12 headers and libraries are part of the Windows 10 SDK. There is no separate download or installation required to use Direct3D 12.

In this section

TOPIC	DESCRIPTION
Direct3D 12 programming guide	Direct3D 12 provides an API and platform that allows apps to take advantage of the graphics and computing capabilities of PCs equipped with one or more Direct3D 12-compatible GPUs.
Direct3D 12 reference	This section covers APIs for Direct3D 12-based graphics programming.
Direct3D 12 glossary	These terms are distinctive of Direct3D 12.

Direct3D 11 Graphics

11/2/2020 • 2 minutes to read • Edit Online

You can use Microsoft Direct3D 11 graphics to create 3-D graphics for games and scientific and desktop applications.

This section contains information about programming with Direct3D 11 graphics.

For more information, see Direct3D 11 Features.

Supported runtime environments	Windows/C++
Recommended programming languages	C/C++
Minimum supported Client	Windows 7
Minimum supported Server	Windows Server 2008 R2

TOPIC	DESCRIPTION
How to Use Direct3D 11	This section demonstrates how to use the Direct3D 11 API to accomplish several common tasks.
What's new in Direct3D 11	This section describes features added in Direct3D 11, Direct3D 11.1, and Direct3D 11.2.
Programming Guide for Direct3D 11	The programming guide contains information about how to use the Direct3D 11 programmable pipeline to create realtime 3D graphics for games as well as scientific and desktop applications.
Direct3D 11 Reference	This section contains the reference pages for Direct3D 11-based graphics programming.

In addition to Direct3D 11 being supported by Windows 7 and later and Windows Server 2008 R2 and later, Direct3D 11 is available down-level for Windows Vista with Service Pack 2 (SP2) and Windows Server 2008 by downloading KB 971644 and KB 971512.

For info about new Direct3D 11.1 features that are included with Windows 8, Windows Server 2012, and are partially available on Windows 7 and Windows Server 2008 R2 via the Platform Update for Windows 7, see Direct3D 11.1 Features and the Platform Update for Windows 7.

DXGI

2/18/2021 • 2 minutes to read • Edit Online

Microsoft DirectX Graphics Infrastructure (DXGI) handles enumerating graphics adapters, enumerating display modes, selecting buffer formats, sharing resources between processes (such as, between applications and the Desktop Window Manager (DWM)), and presenting rendered frames to a window or monitor for display.

DXGI is used by Direct3D 10, Direct3D 11 and Direct3D 12.

Though most graphics programming is done using Direct3D, you can use DXGI to present frames to a window, monitor, or other graphics component for eventual composition and display. You can also use DXGI to read the contents on a monitor.

This documentation set contains information about programming with DXGI.

Supported runtime environments	Windows/C++
Recommended programming languages	C/C++
Minimum supported Client	Windows Vista
Minimum supported Server	Windows Server 2008

- Programming Guide for DXGI
- DXGI Reference

High-level shader language (HLSL)

2/18/2021 • 2 minutes to read • Edit Online

HLSL is the C-like high-level shader language that you use with programmable shaders in DirectX.

For example, you can use HLSL to write a vertex shader, or a pixel shader, and use those shaders in the implementation of the renderer in your Direct3D application.

Or you could use HLSL to write a compute shader, perhaps to implement a physics simulation. However, if for example you're inclined to write your own convolution operator (for image processing) as HLSL in a compute shader, then you'll get better performance in that scenario if you use Direct Machine Learning (DirectML) instead.

HLSL was created (starting with DirectX 9) to set up the programmable 3D pipeline. You can program the entire pipeline with HLSL instructions.

Where to go next

- Programming guide for HLSL
- Reference for HLSL

Programming guide for HLSL

For a conceptual introduction to HLSL, see the Programming guide for HLSL.

The programming guide discusses the different kinds of shader stages, and how to create, compile, optimize, bind, and link shaders.

There you'll also find overviews of, and release notes about, the successive generations of shader model version that have been released, going back as far as HLSL shader model 5.

Reference for HLSL

For HLSL reference documentation, see the Reference for HLSL.

The reference section has a complete listing of the language syntax and of the intrinsic functions that are built into HLSL in order to simplify your coding requirements.

There also you'll find a discussion of shader models versus profiles, and shader model reference content going back as far as HLSL shader model 1. There's also content covering assembly instructions, the D3DCompiler tool, and info about the errors and warnings that a shader can return.

DDS

11/2/2020 • 2 minutes to read • Edit Online

The DirectDraw Surface file format (.dds) was introduced with DirectX 7 to store uncompressed and compressed (DXTn) textures. The file format supports mipmaps, cube maps, and volume maps. The DDS file format is supported by DirectXTex, DirectXTK, legacy D3DX, and other DirectX tools. Starting with Direct3D 10, DDS files support texture arrays.

NOTE

The D3DX (D3DX 9, D3DX 10, and D3DX 11) utility library is deprecated for Windows 8 and is not supported for Windows Store apps. We recommended that you make use of DirectXTex, DirectXTK, or both.

- Programming Guide for DDS
- Reference for DDS

DXCore

2/18/2021 • 2 minutes to read • Edit Online

DXCore is an adapter enumeration API for graphics and compute devices, so some of its facilities overlap with those of DXGI. DXCore is used by Direct3D 12.

This documentation set contains information about programming with DXCore, as well as the DXCore reference.

Requirements

Supported runtime environments	Windows/C++
Recommended programming languages	C++
Minimum supported client	Windows 10, version 2004 (10.0; Build 19041)
Header	dxcore.h and dxcore_interface.h
Library	dxcore.lib
DLL	dxcore.dll

In this section

TOPIC	DESCRIPTION
DXCore programming guide	Guidance for programming with DXCore.
DXCore reference	This section covers DXCore APIs declared in dxcore.h and dxcore_interface.h.

DirectWrite (DWrite)

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Today's applications must support high-quality text rendering, resolution-independent outline fonts, and full Unicode text and layout support. DirectWrite, a DirectX API, provides these features and more.

- A device-independent text layout system that improves text readability in documents and in UI.
- High-quality, sub-pixel, Microsoft ClearType text rendering that can use GDI, Direct2D, or application-specific rendering technology.
- Hardware-accelerated text, when used with Direct2D.
- Support for multi-format text.
- Support for the advanced typography features of OpenType fonts.
- Support for the layout and rendering of text in all supported languages.
- GDI-compatible layout and rendering.

The API supports measuring, drawing, and hit-testing of multi-format text. DirectWrite handles text in all supported languages for global and localized applications, building on the key language infrastructure found in Windows 7. DirectWrite also provides a low-level glyph rendering API for developers who want to perform their own layout and Unicode-to-glyph processing.

NOTE

Project Reunion introduces a new version of DirectWrite—called DWriteCore—that runs on versions of Windows down to Windows 8, and opens the door for you to use it cross-platform. For more details, see DWriteCore overview.

Run-time requirements

- Windows 7 or Windows Vista with Service Pack 2 (SP2) and Platform Update for Windows Vista
- Windows Server 2008 R2 or Windows Server 2008 with Service Pack 2 (SP2) and Platform Update for Windows Server 2008

In this section

TOPIC	DESCRIPTION
What's new in DirectWrite	Here are some of the new additions to DirectWrite.
Programming Guide	The following topics provide an overview of the DirectWrite API.
API Reference	Describes the DirectWrite API.
Sample Code	This section contains information about sample programs for DirectWrite.

DirectXMath

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The DirectXMath API provides SIMD-friendly C++ types and functions for common linear algebra and graphics math operations common to DirectX applications. The library provides optimized versions for Windows 32-bit (x86), Windows 64-bit (x64), and Windows on ARM/ARM64 through SSE, AVX, and ARM-NEON intrinsics support in the Visual C++ compiler.

For developers new to DirectXMath, you may want to consider using the SimpleMath wrapper in the *DirectX Tool Kit* for DirectX 11 / DirectX12 as a starting point.

In this section

TOPIC	DESCRIPTION
DirectXMath Programming Guide	DirectXMath provides a math solution optimized for Windows.
DirectXMath Programming Reference	This section contains reference material for the DirectXMath Library.

Developer audience

The DirectXMath library is designed for C++ developers working on games and DirectX graphics in Windows Store apps, Xbox games, and traditional desktop apps for Windows.

Obtaining DirectXMath

The DirectXMath headers ship in the Windows SDK that comes with Visual Studio 2012 or later, and as an all inline header there is no DLL or static library to link against. It is also available as a package on NuGet.

DirectXMath is open source under the MIT license hosted on GitHub.

windowsnumerics.h APIs

11/2/2020 • 2 minutes to read • Edit Online

The windowsnumerics.h header file defines C++ vector and matrix types in the Windows.Foundation.Numerics namespace. It extends the structs from **Windows.Foundation.Numerics** with a range of mathematical operators and functions.

This namespace is available only in C++. Its .NET equivalent is System.Numerics.

In this section

TOPIC	DESCRIPTION
float2 structure	A vector with two components.
float3 structure	A vector with three components.
float3x2 structure	A 3x2 matrix, used for 2D transforms.
float4 structure	A vector with four components.
float4x4 structure	A 4x4 matrix, used for 3D transforms.
plane structure	This structure represents a plane using a 3D vector normal and a distance value.
quaternion structure	A four dimensional vector, used to represent a rotation.
Windows numerics and DirectXMath interop APIs	These functions convert Windows.Foundation.Numerics types to and from the DirectXMath SIMD types XMVECTOR and XMMATRIX.

DirectX graphics and gaming

2/18/2021 • 2 minutes to read • Edit Online

This content focuses on using DirectX in a Win32 application. For information on using DirectX in a UWP application, see the Windows 10 game development guide (UWP).

In this section

TOPIC	DESCRIPTION
Getting Started with DirectX graphics	Microsoft DirectX graphics provides a set of APIs that you can use to create games and other high-performance multimedia apps. DirectX graphics includes support for high-performance 2-D and 3-D graphics.
Programming DirectX with COM	The Microsoft Component Object Model (COM) is an object-oriented programming model used by several technologies, including the bulk of the DirectX API surface.
Direct2D	Direct2D is a hardware-accelerated, immediate-mode, 2-D graphics API that provides high performance and high-quality rendering for 2-D geometry, bitmaps, and text.
Direct3D	Direct3D enables you to create 3-D graphics for games and scientific apps.
DXCore	DXCore is an adapter enumeration API for graphics and compute devices, so some of its facilities overlap with those of DXGI.
DirectWrite	DirectWrite supports high-quality text rendering, resolution-independent outline fonts, and full Unicode text and layouts.
DirectXMath	DirectXMath provides an optimal and portable interface for arithmetic and linear algebra operations on single-precision floating-point vectors (2D, 3D, and 4D) or matrices (3×3 and 4×4).
windowsnumerics.h APIs	The `windowsnumerics.h` header file defines C++ vector and matrix types in the Windows.Foundation.Numerics namespace.
Classic DirectX Graphics	Microsoft DirectX graphics technologies that are currently minimally used. We do not recommend using these classic DirectX graphics technologies for new apps.
Tools for DirectX Graphics	Describes tools for DirectX Graphics.
DirectX Graphics Articles	Contains technical articles for DirectX Graphics.
XAudio2 APIs	Provides a signal processing and mixing foundation for games. XAudio2 replaces DirectSound.

TOPIC	DESCRIPTION
XInput game controller APIs	Describes how to use the XInput API to interact with the Xbox 360 Controller when it is connected to a Windows computer. XInput replaces DirectInput.

Related topics

- Audio and Video
- Graphics and Gaming

Direct3D 10 Graphics

2/18/2021 • 2 minutes to read • Edit Online

This section contains information about programming with Microsoft Direct3D 10.

In This Section

ITEM	DESCRIPTION
Programming Guide for Direct3D 10	This section contains architecture descriptions, functional block diagrams, descriptions of the building blocks in the pipeline, and code snippets.
Reference for Direct3D 10	This section contains the reference pages for the graphics components including the syntax for the API methods, functions, and data structures.

Reference for Direct3D 9

2/18/2021 • 2 minutes to read • Edit Online

This section contains the reference pages for the Direct3D API. Information is contained in the following sections:

- Direct3D Reference
- D3DX Reference
- Effect Reference
- X File Reference

Related topics

Direct3D 9 Graphics

DirectDraw

2/22/2020 • 2 minutes to read • Edit Online

This section contains information about programming with the DirectDraw component of the DirectX application programming interface (API).

DirectDraw is no longer recommended for use. With the release of Direct3D 9.0, all two-dimensional functionality is contained within Direct3D, its associated helper functions in D3DX, and the DirectX 11 technology Direct2D. However, the DirectDraw reference documentation is still available in this section.

In This Section

This section contains the DirectDraw Reference documentation.

DirectX Technical Articles

11/2/2020 • 5 minutes to read • Edit Online

This section contains a series of technical articles about developing games for Microsoft Windows.

In this section

TOPIC	DESCRIPTION
64-bit programming for Game Developers	This article addresses compatibility and porting issues and helps developers ease their transition to 64-bit platforms.
Authenticode Signing for Game Developers	This article discusses how to get started with authenticating your game and how to integrate authentication into a daily build process.
Best Security Practices in Game Development	This article discusses best practices to use in game development.
Cascaded Shadow Maps	Cascaded shadow maps (CSMs) are the best way to combat one of the most prevalent errors with shadowing: perspective aliasing.
Coding For Multiple Cores on Xbox 360 and Microsoft Windows	This topic gives some advice on how to get started with multithreaded programming.
Common Techniques to Improve Shadow Depth Maps	This technical article provides an overview of some common shadow depth map algorithms and common artifacts, and explains several techniques ranging in difficulty from basic to intermediate that can be used to increase the quality of standard shadow maps.
Crash Dump Analysis	This technical article provides info about how to write and use a minidump.
Debugging with Symbols	This article provides a high level overview of how to best use symbols in your debugging process.
Direct3D 10 Frequently Asked Questions	This article contains some of the frequently asked questions surrounding Direct3D 10 from the point of view of a developer who is porting an existing application from Direct3D 9 (D3D9) to Direct3D 10 (D3D10).
DirectX Frequently Asked Questions	This article contains a collection of Frequently Asked Questions (FAQ) about Microsoft DirectX.
DirectX Installation for Game Developers	This article is intended to address some of the common questions about the DirectX runtime, and using DirectSetup to install DirectX.

TOPIC	DESCRIPTION
Disabling Shortcut Keys in Games	This articles describes how to temporarily disable keyboard shortcuts in Microsoft Windows to prevent disruption of game play for full screen games.
Game Timing and Multicore Processors	This article suggests a more accurate, reliable solution to obtain high-resolution CPU timings by using the Windows APIs **QueryPerformanceCounter** and **QueryPerformanceFrequency**.
Games for Windows Technical Requirements	This article provides technical requirements and best practices for games that run on Windows.
Games for Windows Test Cases	This article provides test cases for games for Windows.
Gaming with Least-Privileged User Accounts	This article describes how games developers can author Microsoft Windows games that work well with least-privileged user accounts (also known as limited-user accounts).
Install-on-Demand for Games	This technical article discusses two techniques, install-on-demand and background install, using Windows Installer.
Installation and Maintenance of Games	This article describes a set of best practices that can help reduce user frustration about the time required to install a game, prevent unnecessary support calls, and allow users to start playing your game as quickly and painlessly as possible.
Lockless Programming Considerations for Xbox 360 and Microsoft Windows	This article gives an overview of some of the issues to consider when trying to use lockless programming techniques.
Installation Best Practices for Massively Multiplayer Online Games	This article describes creating a chain of trust design for Massively Multiplayer Online Games (MMOG) client installation and custom game update systems that work well with Windows and the security model of Windows Vista and Windows 7.
Installing and Using Input Method Editors	This article offers a tutorial for how to install and use the standard Windows Input Method Editor (IME).
Introduction to the 10-Foot Experience for Windows Game Developers	This article introduces the 10-foot experience and explores the list of things that you should consider first about this new interaction pattern, even if you aren't expecting your game to be played this way.
Making Video Games Accessible: Business Justifications and Design Considerations	This article is for game content developers and producers who want to reach the accessibility community market by adding basic accessibility features to help people with disabilities or impairments.
Optimizing DVD performance for Windows Games	This article discusses how to optimize DVD performance for Windows games.

TOPIC	DESCRIPTION
Patching Game Software in Windows XP, Windows Vista, and Windows 7	This article examines some methods of patching that will work well in Windows Vista and Windows 7, as well as Windows XP.
Resource Management Best Practices	This article discusses best practices for dealing with resources generally, how managed and unmanaged resources behave, and provides some detail on how resources are typically handled by the runtime and drivers.
Simplifying Game Installation	This article outlines some concepts that developers of games for Windows can and should implement to improve the overall experience.
Taking Advantage of High-Definition Mouse Movement	This article focuses on the best way to optimize the performance of high-definition mouse input in a game like a first-person shooter.
The Direct3D Transformation Pipeline	This article provides a technical explanation for Direct3D application developers on how to set the parameters of the Direct3D Transformation Pipeline by the direct manipulation of Direct3D matrices.
Top Issues for Windows Titles	This article highlights many of the common issues we've seen in current-generation PC games.
Top Tools and Techniques for Making More Robust Windows Games	This articles describes tools and techniques that you can use to help reduce the number of support calls you receive.
User Account Control for Game Developers	This article describes the guidelines and best practices for game developers to work effectively with the User Account Control (UAC) security feature introduced in Windows Vista.
Using an Input Method Editor in a Game	This article explains how you can implement a basic IME edit control in a full-screen DirectX application.
Windows Firewall for Game Developers	This article describes the Windows Firewall - why it exists, what it accomplishes, and how it does so. Most importantly, it describes how to configure your game to work well with the firewall.
Windows Games Explorer for Game Developers	This article outlines the process of registering a game with Games Explorer and parental controls on Windows Vista and Windows 7 by using the new GDF schema.
Windows Installer for Game Developers	This article gives an overview of Windows Installer, specifically targeted to game developers. For detailed documentation on the features and APIs mentioned in this article, please refer to the Windows Platform SDK.

Where is the DirectX SDK?

2/18/2021 • 10 minutes to read • Edit Online

Starting with Windows 8, the DirectX SDK is included as part of the Windows SDK.

We originally created the DirectX SDK as a high-performance platform for game development on top of Windows. As DirectX technologies matured, they became relevant to a broader range of applications. Today, the availability of Direct3D hardware in computers drives even traditional desktop applications to use graphics hardware acceleration. In parallel, DirectX technologies are more integrated with Windows. DirectX is now a fundamental part of Windows.

Because the Windows SDK is the primary developer SDK for Windows, DirectX is now included in it. You can now use the Windows SDK to build great games for Windows. To download the Windows 8.x SDK or Windows 10 SDK, see Windows SDK and emulator archive.

The following technologies and tools, formerly part of the DirectX SDK, are now part of the Windows SDK.

TECHNOLOGY OR TOOL	DESCRIPTION
Windows Graphics Components	The headers and libraries for Direct3D and other Windows graphics APIs, like Direct2D, are available in the Windows SDK. [!Note] D3DX9/D3DX10/D3DX11 was only available in the legacy DirectX SDK. The D3DCSX DirectCompute utility library and redistributable DLL is available in the Windows SDK. D3DX12 is available on GitHub.
HLSL compiler (FXC.EXE)	The HLSL compiler is a tool in the appropriate architecture subdirectory under the bin folder in the Windows SDK. [!Note] The D3DCompiler API and redistributable DLL is available in the Windows SDK.
PIX for Windows	A replacement for the PIX for Windows tool is now a feature in Microsoft Visual Studio, called Visual Studio Graphics Debugger. This feature has greatly improved usability, support for Windows 8, and Direct3D 11.1, and integration with traditional Microsoft Visual Studio features such as call stacks and debugging windows for HLSL debugging. For more info about this new feature, see Debugging DirectX Graphics. For DirectX 12 development, see the latest generation of PIX on Windows

TECHNOLOGY OR TOOL	DESCRIPTION
XAudio2 for Windows	The XAudio2 API is now a system component in Windows 8.x and Windows 10. The headers and libraries for XAudio2 are available in the Windows SDK. For Windows 7 support, see XAudio2Redist.
XInput for Windows	The XInput 1.4 API is now a system component in Windows 8.x and Windows 10. The headers and libraries for XInput are available in the Windows SDK. > [!Note] > Legacy XInput 9.1.0 is also available as part of Windows 7 or later.
XNAMATH	The most recent version of XNAMATH, which is updated for new instruction sets as well as ARM/ARM64, is now DirectXMath. The headers for DirectXMath are available in the Windows SDK and on GitHub.
DirectX Control Panel and DirectX Capabilities Viewer	The DirectX Control Panel and DirectX Capabilities Viewer utilities are included in the appropriate architecture subdirectory under the bin folder in the Windows SDK. DirectX Capabilities Viewer is also available on GitHub.
XACT	The Xbox Audio Cross Platform Tool (XACT) is no longer supported for use on Windows.
Games Explorer and GDFMAKER	The Games Explorer API presents games to users of Windows. The Games Explorer API is supported only on Windows Vista and Windows 7. Use the Games Definition File Maker tool (GDFMAKER.EXE) to declare game ratings for Windows Store apps. The Game Definition File Maker tool (GDFMaker.exe) is included in the x86 subdirectory under the bin folder in the Windows SDK, and supports both Windows Store apps and Win32 desktop applications.
Samples	You can find sample applications that highlight DirectX technologies on Windows in the DirectX samples repo, and in the Code Gallery archive. Most samples for older versions of Direct3D are available only for download in previous versions of the DirectX SDK, although a number of them are online as well. For more info about these samples, see DirectX SDK Samples Catalog.
Managed DirectX 1.1	The .NET DirectX assemblies are deprecated and are not recommended for use by new applications. There are a number of alternatives available. See DirectX and .NET.

The legacy DirectX SDK is no longer available for download from *Microsoft Download Center* per the retirement of all SHA-1 signed content.

NOTE

The DirectX SDK fails to install if you have a certain version of the Visual C++ 2010 Redistributable Package already installed. For more info about and a solution to fix this issue, see "S1023" error when you install the DirectX SDK (June 2010).

Using DirectX SDK projects with Visual Studio

The samples from the June 2010 DirectX SDK are supported with premium Visual Studio SKUs (Microsoft Visual Studio Professional 2012, Microsoft Visual Studio Ultimate 2012, Microsoft Visual Studio Professional 2013, or Microsoft Visual Studio Ultimate 2013) on Windows 7 and the Windows 8 and later releases. Due to the transition of DirectX headers and libraries into the Windows SDK, changes to the project settings are needed to build these samples correctly with how the Windows 8 SDK and later is packaged with the premium Visual Studio SKUs.

These steps also apply to your own projects that are dependent on the DirectX SDK.

1. Ensure that the June 2010 release of the DirectX SDK is installed on your development computer. If you install onto a computer running Windows 8 and later, you will be prompted and required to enable .NET 3.5 as a prerequisite installation to the DirectX SDK.

   NOTE

   The DirectX SDK fails to install if you have a certain version of the Visual C++ 2010 Redistributable Package already installed. For more info about and a solution to fix this issue, see "S1023" error when you install the DirectX SDK (June 2010).

2. Make sure that you are using one of the premium Visual Studio SKUs. Microsoft Visual Studio Express 2012 for Windows 8 or Microsoft Visual Studio Express 2013 for Windows won't build Windows 8 and later desktop applications such as the DirectX SDK samples. To install one of the premium Visual Studio SKUs, go to: Visual Studio downloads and follow the instructions.

3. Use the DirectX SDK Sample Browser to install the project files for the desired sample. Open the sample's Microsoft Visual Studio 2010 compatible solution file (suffixed with **_2010**).

4. If you are opening the sample on a system that only has Microsoft Visual Studio 2012 or Microsoft Visual Studio 2013 installed, you get the following message: "This solution contains one or more projects using an earlier version of VC++ compiler and libraries. Each project can be updated to use the VC++ compiler and libraries (v110)." Choose the **Update** option from this dialog box to update before you open the project.

   Otherwise, you can update to the Visual Studio 2012 or Visual Studio 2013 C++ 11 compiler and libraries after they have loaded by right-clicking on the solution and choosing **Update VC++ projects**.

5. D3DX is not considered the canonical API for using Direct3D in Windows 8 and later and therefore isn't included with the corresponding Windows SDK. Investigate alternate solutions for working with the Direct3D API. For legacy projects, such as the Windows 7 (and earlier) DirectX SDK samples, the following steps are necessary to build applications with D3DX using the DirectX SDK:

   a. Modify the project's **VC++** directories as follows to use the right order for SDK headers and libraries.

i. Open **Properties** for the project and select the **VC++ Directories** page. ii. Select **All Configurations and All Platforms**. iii. Set these directories as follows:

- Executable Directories: (On right-side drop-down)
- Include Directories: **$(IncludePath);$(DXSDK_DIR)Include**
- Include Library Directories: **$(LibraryPath);$(DXSDK_DIR)Lib\x86**

iv. Click **Apply**.
v. Choose the **x64 Platform**.
vi. Set the **Library Directory** as follows:

- Library Directories: **$(LibraryPath);$(DXSDK_DIR)Lib\x64**

b. Wherever "d3dx9.h", "d3dx10.h", or "d3dx11.h" are included in your project, be sure to explicitly include "d3d9.h", "d3d10.h" and "dxgi.h", or "d3d11.h" and "dxgi.h" first to ensure you are picking up the newer version. You can disable **warning C4005** if needed; however, this warning indicates you are using the older version of these headers.

c. Remove all references to DXGIType.h in your project. This header doesn't exist in the Windows SDK, and the DirectX SDK version conflicts with the new winerror.h.

d. All D3DX DLLs are installed onto your development computer by the DirectX SDK installation. Ensure that the necessary D3DX dependencies are redistributed with any sample or with your application if it is moved to another machine.

e. Be aware that replacement technologies for current uses of D3DX11 include DirectXTex, DirectXTK, DirectXMesh, and UVAtlas. D3DXMath is replaced by DirectXMath.

6. Ensure that you are using the new version of the HLSL shader compiler by observing the following conditions:

   a. Changing the executable directory as per step 5 will cause project builds to use FXC from the Windows SDK install. Be aware that HLSL files are now officially recognized by Visual Studio. You can add them as project files and set compiler options through the project system.

   b. Invoking run-time compilation through the legacy D3DX DLL will use the incorrect older version of the HLSL compiler. Replace all references to D3DXCompile*, D3DX10Compile*, and D3DX11Compile* APIs in your code with the D3DCompile function in D3DCOMPILER_46.DLL or D3DCOMPILER_47.DLL.

   c. Any project that uses run-time shader compilation must have D3DCOMPILER_xx.DLL copied to the local executable path for the project. This DLL is available in this sub-directory of the Windows SDK installation under **%ProgramFiles(x86)%\Windows Kits\8.0\Redist\D3D\** or **%ProgramFiles(x86)%\Windows Kits\8.1\Redist\D3D\** where is **x86** and **x64**.

   The D3DCOMPILER_46.DLL or D3DCOMPILER_47.DLL from the Windows SDK is not a system component and should not be copied to the Windows system directory. You can redistribute this DLL to other computers with your application as a side-by-side DLL.

7. Any project that uses the XInput API and is intended to run on Windows 7 or older versions of Windows need to use either the legacy version (9.1.0) or will need to explicitly include the headers and libraries for this component from the DirectX SDK. The XInput header and XINPUT.LIB that are included in the Windows SDK target only the version (1.4) that ship as part of Windows 8 and later. The same header can be used with XINPUT9_1_0.LIB to use the legacy version, which is included with older versions of Windows. The legacy version of XInput doesn't detect full capabilities or support controller-integrated audio, so if support for these features is required, you must use the DirectX SDK version (1.3).

   To use the full-featured down-level XInput API, you should `#include` the specific XInput headers from the

DirectX SDK directly:

```
#include <%DXSDK_DIR%Include\xinput.h>
```

...and in your linker options for Additional Dependencies, link directly to the DirectX SDK XInput library:

%DXSDK_DIR%Include\\xinput.lib

The XINPUT1_3.DLL binary is installed to the Windows system directories by the DirectX SDK installation on your development computer. You will need to redistribute this binary with your application using the DirectX Setup installation from the DirectX SDK.

8. Any project that uses the XAudio2 API and is intended to run on Windows 7 or older versions of Windows need to use either the older version (9.1.0) or explicitly include the headers and libraries for this component from the DirectX SDK. The XAudio2 headers and libraries that are included with the Windows SDK target only the version (2.8) that is included as part of Windows 8.

   For example, with XAudio2, you should `#include` the specific XAudio2 headers from the DirectX SDK directly:

   ```
   #include <%DXSDK_DIR%Include\xaudio2.h>
   ```

   ...and in your linker options for Additional Dependencies, link directly to the DirectX SDK XAudio2 library:

   %DXSDK_DIR%Include\\xaudio2.lib

   The XAUDIO2_7.DLL binary is installed to the Windows system directories by the DirectX SDK installation on your development computer. You need to redistribute these libraries with your application using the DirectX Setup installation from the DirectX SDK.

9. If you've used the DirectX SDK with past versions of Visual Studio, the Visual Studio 2010 upgrade might have migrated the DirectX SDK path into your default project settings. It is recommended that you remove these settings to prevent future build errors. In the **%USERPROFILE%\AppData\Local\Microsoft\MSBuild\v4.0** directory, modify the **Microsoft.Cpp.Win32.user** and **Microsoft.Cpp.x64.user** files to remove all references to DXSDK_DIR paths. Alternatively, you can remove the entire node that contains the Path entries such as and to revert to standard defaults. If you don't see references to DXSDK_DIR in these files, no changes are necessary.

10. If the resulting app supports Windows Vista with Service Pack 2 (SP2) as well as Windows 7 and Windows 8 and later, set the Preprocessor Definition named _**WIN32_WINNT** to 0x600. If it only supports Windows 7 and Windows 8 and later, set it to 0x601.

    For example:

    a. Open **Properties** for the project and select **C/C++** > **Preprocessor**.
    b. Select **All Configurations** and **All Platforms**.
    c. Go to the **Preprocessor Definitions** section and set _WIN32_WINNT=0x600.
    d. Click **Apply**.

Related topics

Games for Windows and the DirectX SDK

Where is the DirectX SDK (2021 Edition)?

DirectX SDKs of a certain age

Living without D3DX

DirectX Graphics Articles

11/2/2020 • 2 minutes to read • Edit Online

Purpose

This section contains technical articles that describe WARP and Windows 7's newly added support for Flip Mode Present and its associated Present Statistics in Direct3D 9Ex and Desktop Window Manager. This section also contains technical articles about Windows graphics APIs, porting Direct3D 9 APIs to Microsoft DirectX Graphics Infrastructure (DXGI) APIs, and how to deploy Direct3D 11.

For more information about Direct3D, see Direct3D.

Developer audience

These technical articles are for DirectX graphics application developers.

In this section

TOPIC	DESCRIPTION
Platform Update for Windows 7	Describes what components of the Windows 8 graphics stack become available, and to what extent, through the Platform Update for Windows 7.
Direct3D 9Ex Improvements	Describes Windows 7's newly added support for Flip Mode Present and its associated Present Statistics in Direct3D 9Ex and Desktop Window Manager. Target applications include video or framerate-based presentation applications. Applications using Direct3D 9Ex Flip Mode Present reduce the system resource load when DWM is enabled. Present Statistics enhancements associated with Flip Mode Present enable Direct3D 9Ex applications to better control the rate of presentation by providing real-time feedback and correction mechanisms. Detailed explanations and pointers to sample resources are included.
Surface Sharing Between Windows Graphics APIs	Provides a technical overview of interoperability using surface sharing between Windows graphics APIs, including Direct3D 11, Direct2D, Direct3D 10, and Direct3D 9Ex. If you already have a working knowledge of these APIs, this paper can help you use multiple APIs to render to the same surface in an application designed for the Windows 7 or Windows Vista operating systems. This topic also provides best practice guidelines and pointers to additional resources.
WARP Guide	Provides a guide of Windows Advanced Rasterization Platform (WARP), a high speed software rasterizer.
Graphics APIs in Windows	Discusses Windows graphics features and APIs.
Direct3D 11 Deployment for Game Developers	Describes how to deploy the Direct3D 11 components on a system.

TOPIC	DESCRIPTION
DirectX Graphics Infrastructure (DXGI): Best Practices	Discusses issues about porting Direct3D 9 APIs to DXGI APIs and features of various versions of DXGI.
Using DirectX with high dynamic range displays and advanced color	This topic provides a technical overview of rendering high dynamic range Direct3D 11 and Direct3D 12 content to an HDR10 display using Windows 10 advanced color support.

XAudio2 APIs

2/18/2021 • 2 minutes to read • Edit Online

Purpose

XAudio2 is a low-level audio API that provides signal processing and mixing foundation for developing high performance audio engines for games.

In this section

TOPIC	DESCRIPTION
Programming Guide	This section lists the overview topics for the XAudio2 application programming interface (API).
Programming Reference	This section contains reference topics for the Microsoft XAudio2 application programming interface (API).

Developer audience

XAudio2 APIs are designed for use by game developers who want to improve performance in their games.

XInput Game Controller APIs

2/22/2020 • 2 minutes to read • Edit Online

Purpose

XInput Game Controller API enables applications to receive input from the Xbox Controller for Windows.

In this section

TOPIC	DESCRIPTION
Programming Guide	This guide contains information on how to use the XInput API to interact with the Xbox Controller when it is connected to a Windows PC.
Programming Reference	XInput functions and structures.

Developer audience

XInput Game Controller APIs is designed for use by developers who want to use the Xbox Controller for their Windows applications.

DirectComposition

2/22/2020 • 2 minutes to read • Edit Online

NOTE

For apps on Windows 10, we recommend using Windows.UI.Composition APIs instead of DirectComposition. For more info, see Modernize your desktop app using the Visual layer.

Purpose

Microsoft DirectComposition is a Windows component that enables high-performance bitmap composition with transforms, effects, and animations. Application developers can use the DirectComposition API to create visually engaging user interfaces that feature rich and fluid animated transitions from one visual to another.

DirectComposition enables rich and fluid transitions by achieving a high framerate, using graphics hardware, and operating independently of the UI thread. DirectComposition can accept bitmap content drawn by different rendering libraries, including Microsoft DirectX bitmaps, and bitmaps rendered to a window (HWND bitmaps). Also, DirectComposition supports a variety of transformations, such as 2D affine transforms and 3D perspective transforms, as well as basic effects such as clipping and opacity.

DirectComposition is designed to simplify the process of composing *visuals* and creating animated transitions. If your application already contains rendering code or already uses the recommended DirectX API, you only need to do a minimal amount of work to use DirectComposition effectively.

Developer audience

The DirectComposition API is intended for experienced and highly-capable graphics developers who know C/C++, have a solid understanding of the Component Object Model (COM), and are familiar with Windows programming concepts.

Run-time requirements

DirectComposition was introduced in Windows 8. It is included in 32-bit, 64-bit, and ARM platforms.

In this section

TOPIC	DESCRIPTION
Why use DirectComposition?	This topic describes the capabilities and benefits of DirectComposition.
How to Use DirectComposition	This section describes best practices for using the DirectComposition API, and demonstrates how to use the API to accomplish several common tasks.
DirectComposition concepts	This section provides a conceptual overview of DirectComposition.
DirectComposition reference	This section provides detailed reference information for the elements that make up the DirectComposition API.

TOPIC	DESCRIPTION
DirectComposition samples	The following sample applications show how to use the DirectComposition API and demonstrate its capabilities.
DirectComposition glossary	This topic defines DirectComposition terms.

GDI+

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Windows GDI+ is a class-based API for C/C++ programmers. It enables applications to use graphics and formatted text on both the video display and the printer. Applications based on the Microsoft Win32 API do not access graphics hardware directly. Instead, GDI+ interacts with device drivers on behalf of applications. GDI+ is also supported by Microsoft Win64.

Where applicable

GDI+ functions and classes are not supported for use within a Windows service. Attempting to use these functions and classes from a Windows service may produce unexpected problems, such as diminished service performance and run-time exceptions or errors.

NOTE

When you use the GDI+ API, you must never allow your application to download arbitrary fonts from untrusted sources. The operating system requires elevated privileges to assure that all installed fonts are trusted.

Developer audience

The GDI+ C++ class-based interface is designed for use by C/C++ programmers. Familiarity with the Windows graphical user interface and message-driven architecture is required.

Run-time requirements

GDI+ can be used in all Windows-based applications. GDI+ was introduced in Windows XP and Windows Server 2003. For information about which operating systems are required to use a particular class or method, see the More Information section of the documentation for the class or method.

In this section

TOPIC	DESCRIPTION
Overview	General information about GDI+.
Using	Tasks and examples using GDI+.
Reference	Documentation of GDI+ C++ class-based API.

Related topics

Windows GDI

DirectX

Windows Image Acquisition

OpenGL

Windows Multimedia

Windows GDI

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Microsoft Windows graphics device interface (GDI) enables applications to use graphics and formatted text on both the video display and the printer. Windows-based applications do not access the graphics hardware directly. Instead, GDI interacts with device drivers on behalf of applications.

Where applicable

GDI can be used in all Windows-based applications.

Developer audience

This API is designed for use by C/C++ programmers. Familiarity with the Windows message-driven architecture is required.

Run-time requirements

For information on which operating systems are required to use a particular function, see the Requirements section of the documentation for the function.

In this section

- Bitmaps
- Brushes
- Clipping
- Colors
- Coordinate Spaces and Transformations
- Device Contexts
- Filled Shapes
- Fonts and Text
- Lines and Curves
- Metafiles
- Multiple Display Monitors
- Painting and Drawing
- Paths
- Pens
- Printing and Print Spooler
- Rectangles
- Regions

Related topics

DirectX

GDI+

OpenGL

Windows Image Acquisition

Monitor Configuration (Monitor Configuration)

2/18/2021 • 2 minutes to read • Edit Online

You can use the monitor configuration functions to get information from a monitor and to change a monitor's settings. You can use these functions to :

- Change a monitor's geometry settings, such as the width and height of the display.
- Change image settings, such as brightness and contrast.
- Reset a monitor's settings to their factory defaults.
- Degauss the monitor.

Internally, the monitor configuration functions use the Display Data Channel Command Interface (DDC/CI) to send commands to the monitor.

The monitor configuration functions are documented in the following sections:

- About Monitor Configuration
- Using Monitor Configuration
- Monitor Configuration Reference

OpenGL

11/2/2020 • 2 minutes to read • Edit Online

Purpose

As a software interface for graphics hardware, OpenGL renders multidimensional objects into a framebuffer. The Microsoft implementation of OpenGL for the Windows operating system is industry-standard graphics software with which programmers can create high-quality still and animated three-dimensional color images. The version of OpenGL described in this section is 1.1.

For information about OpenGL ES running on Windows, see ANGLE for Windows Store.

Where applicable

OpenGL is built for compatibility across hardware and operating systems. This architecture makes it easy to port OpenGL programs from one system to another. While each operating system has unique requirements, the OpenGL code in many programs can be used as is.

Developer audience

Designed for use by C/C++ programmers, OpenGL requires familiarity with the Windows graphical user interface as well as message-driven architecture.

Run-time requirements

For more information on which operating systems are required for a particular function, see the Requirements section of the documentation for the function.

In this section

TOPIC	DESCRIPTION
Overview	General information about how OpenGL works.
Reference	Documentation of functions, structures, and other programming elements.
Samples	Examples of OpenGL code.

Related topics

DirectX Graphics and Gaming

Still Image

Windows Color System (WCS)

Windows GDI

Windows Image Acquisition

Windows Multimedia

Windows Imaging Component

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Windows Imaging Component (WIC) is an extensible platform that provides low-level API for digital images. WIC supports the standard web image formats, high dynamic range images, and raw camera data. WIC also supports additional features such as:

- Built-in support for standard metadata formats
- Extensible framework for image codecs, pixel formats, and metadata formats.
- Wide range of pixel format support.
- High-color support; including 30-bit extended range, 30-bit high precision, and 48-bit high precision and wide gamut pixel formats.
- Progressive image decoding.

Developer Audience

WIC is designed to meet the needs of the following classes of developers:

- Developers who read and write image data in an application.
- Developers who read and write image metadata in an application.
- Developers who want run-time codec discovery for newly designed or implemented image codecs.
- Developers designing or implementing new image codecs and metadata formats.

In this section

TOPIC	DESCRIPTION
What's New in WIC	
Programming Guide	Describes how to use the WIC APIs.
Reference	Contains the reference for the WIC APIs.
WIC Samples	Provides samples for WIC.
Glossary	Defines terms that are used in WIC.

Windows Mixed Reality

2/22/2020 • 2 minutes to read • Edit Online

Learn about the Desktop APIs for Windows Mixed Reality devices. For more information, see the Mixed Reality Dev Center or the Windows.Graphics.Holographic Windows Runtime API reference content.

In this section

TOPIC	DESCRIPTION
Windows Mixed Reality COM Interfaces	The topics contained in this section provide the reference specifications for Windows Mixed Reality interfaces.

Windows Color System

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Color management schemes are implemented in Microsoft Windows operating systems to improve the rendering of color content in all forms of digital communication.

The color management scheme that's used starting with Windows 2000, Windows XP, and Windows Server 2003 is called Image Color Management (ICM) 2.0. The color management scheme that's used starting with Windows Vista is called Windows Color System (WCS) 1.0. The WCS 1.0 color management scheme is a superset of ICM 2.0 APIs and functionality.

Where applicable

ICM can be used in all applications on Windows 2000 and later operating systems. WCS can be used in all applications on Windows Vista and later operating systems.

Developer audience

The WCS API is designed for use by C/C++ programmers. Familiarity with the Windows graphical user interface, message-driven architecture, and a working knowledge of color management concepts are required.

Run-time requirements

Applications that use the ICM API require Windows 2000 or later. Applications that use the WCS API require Windows Vista or later. For specific run-time information on a function, see the Requirements section of the reference page for that function.

In this section

- Security Considerations: Windows Color System
- Basic color management concepts
- Windows Color System Schemas and Algorithms
- About Windows Color System Version 1.0
- Using WCS 1.0
- Reference
- WCS 1.0 Glossary

Related topics

OpenGL

Windows Multimedia

Windows GDI

Networking and Internet

2/18/2021 • 5 minutes to read • Edit Online

Windows has APIs, components, and services that support your desktop apps' use of networking and the Internet. They provide:

- HTTP APIs.
- Network and network management.
- DNS and DHCP.
- Networking capabilities that are independent of particular network implementations.
- Remote Access Service.
- Telephony and fax.

In this section

TOPIC	DESCRIPTION
Delivery Optimization (DO)	Delivery Optimization is a is a cloud-managed, peer-to-peer client update service that delivers updates to an organization's networked devices. Delivery Optimization allows devices to download updates from alternate sources (such as other peers on the network), in addition to Microsoft servers. Delivery Optimization combines partial bits from local devices, with partial bits from Microsoft servers to update devices in the network environment. The expected result is reduced bandwidth usage, and a faster update process.
Domain Name System (DNS)	Domain Name System (DNS), a locator service in Microsoft Windows, is an industry-standard protocol that locates computers on an IP-based network.
Dynamic Host Configuration Protocol (DHCP)	The Dynamic Host Configuration Protocol (DHCP) Application Programming Interface (API), also referred to as DHCP Client Options, enables Microsoft Windows clients to query specific options from DHCP servers. This enables vendor-specific options exposed through DHCP servers to be queried by Windows clients.
Fax Service	The fax service provides fax functionality for clients on a local area network.
Get Connected Wizard API	The Get Connected Wizard application programming interface (API) allows developers to create network, Internet and virtual private network (VPN) connections, determine whether Internet connectivity is present, and register wizard pages for capturing required user information for specific connection types.
HTTP Server API	The HTTP Server API enables applications to communicate over HTTP without using Microsoft Internet Information Server (IIS).

TOPIC	DESCRIPTION
IP Helper	The Internet Protocol Helper (IP Helper) API enables the retrieval and modification of network configuration settings for the local computer.
Management Information Base	The Management Information Base (MIB) API provides a set of structures used to contain network and network management data for a number of technologies, including Remote Access Routing Services, Internet Protocol (IP) Helper, and Simple Network Management Protocol (SNMP).
Message Queuing (MSMQ)	Message Queuing (MSMQ) technology enables applications running at different times to communicate across heterogeneous networks and systems that may be temporarily offline.
Multicast Address Dynamic Client Allocation Protocol (MADCAP)	The Multicast Address Dynamic Client Allocation Protocol (MADCAP) enables applications to obtain, renew, and release multicast addresses.
Network interfaces	This topic describes high-level network interface concepts on Windows, including the ways they can be identified in code and their properties.
Network List Manager	The Network List Manager API enables applications to retrieve a list of available network connections. Applications can filter networks, based on attributes and signatures, and choose the networks best suited to their task. The Network List Manager infrastructure notifies applications of changes in the network environment, thus enabling applications to dynamically update network connections.
Network Management	The network management functions provide the ability to manage user accounts and network resources.
Network Share Management	Network share management allows applications to manage and monitor communications between Windows clients and servers using the Server Messaging Block (SMB) protocol.
Peer-to-Peer	Peer-to-peer technologies are used to facilitate real-time communication and collaboration across distributed networks.
Quality of Service (QOS)	Quality of Service (QOS), an industry-wide initiative, enables more efficient use of the network.
Remote Procedure Call (RPC)	Microsoft Remote Procedure Call (RPC) defines a powerful technology for creating distributed client/server programs. The RPC run-time stubs and libraries manage most of the processes relating to network protocols and communication. This enables you to focus on the details of the application rather than the details of the network.
Routing and Remote Access Service	Remote Access Service (RAS) can be used to create client applications. These applications display RAS common dialog boxes, manage remote access connections and devices, and manipulate phone-book entries.

TOPIC	DESCRIPTION
Simple Network Management Protocol	The Microsoft Windows implementation of the Simple Network Management Protocol (SNMP) is used to configure remote devices, monitor network performance, audit network usage, and detect network faults or inappropriate access.
SMB Management API	The SMB Management API provides WMI classes and methods to manage shares and share access.
Telephony Application Programming Interfaces (TAPI)	The Microsoft telephony application programming interfaces support the development of communications applications for Windows.
Teredo	Teredo is an IPv6 transition technology that provides address assignment and host-to-host automatic tunneling for unicast IPv6 traffic when IPv6/IPv4 hosts are located behind one or multiple IPv4 network address translators (NATs).
WebSocket Protocol Component API	WebSocket Protocol Component API enables asynchronous, bi-directional communication channels over HTTP that work across existing network intermediaries.
Windows Filtering Platform	Windows Filtering Platform (WFP) is a set of API and system services that provide a platform for creating network filtering applications. The WFP API allows developers to write code that interacts with the packet processing that takes place at several layers in the networking stack of the operating system. Network data can be filtered and modified before it reaches its destination.
Windows Firewall Technologies	Windows Firewall with Advanced Security and related firewall technologies enable developers to share Internet connections, protect connections using a firewall, and provide Network Address Translation (NAT).
Windows Networking (WNet)	The Windows networking (WNet) functions allow you to implement networking capabilities in your application without making allowances for a particular network provider or physical network implementation.
Windows Network Virtualization	Windows Network Virtualization enables customer virtual machine networks to decouple virtual machine networks from physical networks, provides flexibility in virtual machine provisioning, and allows customers to bring their IP addresses and topologies into cloud datacenters.
Windows RSS Platform	The Windows RSS Platform is an API that enables applications to access and manipulate the Common Feed List, a collection of Really Simple Syndication (RSS) feeds to which the user has subscribed.
Windows Sockets 2	Windows Sockets 2 (Winsock) enables programmers to create advanced Internet, intranet, and other network-capable applications to transmit application data across the wire, independent of the network protocol being used.

TOPIC	DESCRIPTION
Windows Web Services API	WWSAPI is a native-code implementation of SOAP which provides core network communication functionality by supporting a set of the WS-* and .NET-* family of protocols.
WebDAV	Web Distributed Authoring and Versioning (WebDAV) is an extension to Hypertext Transfer Protocol (HTTP) that defines how basic file functions such as copy, move, delete, and create are performed by using HTTP.
Windows HTTP Services (WinHTTP)	Microsoft Windows HTTP Services (WinHTTP) provides developers with an HTTP client application programming interface (API) to send requests through the HTTP protocol to other HTTP servers.
XML HTTP Request 2	The XML HTTP Request 2 interfaces allow application to conduct HTTP request operations in multithreaded apartments (MTA) and use callbacks to receive notification of required information during response processing.
Windows Internet (WinINet)	The Microsoft Windows Internet (WinINet) application programming interface (API) enables applications to access standard Internet protocols, such as FTP and HTTP. For ease of use, WinINet abstracts these protocols into a high-level interface.

Related topics

Internet

Wireless Networking

Delivery Optimization (DO)

11/2/2020 • 2 minutes to read • Edit Online

Delivery Optimization is a cloud-managed, peer-to-peer client and a downloader service that delivers updates to an organization's networked devices. Delivery Optimization allows devices to download updates from alternate sources (such as other peers on the network), in addition to Microsoft servers. Delivery Optimization combines partial bits from local devices, with partial bits from Microsoft servers to update devices in the network environment. The expected result is reduced bandwidth usage, and a faster update process.

See Configure Delivery Optimization for Windows 10 updates for more information.

In this section

TOPIC	DESCRIPTION
DO Reference	Reference information for the Delivery Optimization programming interfaces.

Domain Name System

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Domain Name System (DNS), a locator service in Microsoft Windows, is an industry-standard protocol that locates computers on an IP-based network. IP networks, such as the Internet and Windows networks, rely on number-based addresses to process data. Users however, can more easily remember name addresses, so it is necessary to translate user-friendly names (such as www.microsoft.com) into addresses that the network can recognize (such as 207.46.131.137).

Where applicable

Windows and Active Directory use DNS. DNS is the primary locator service for the Internet and Active Directory, and therefore, DNS is considered a base service for Windows and Active Directory.

Developer audience

Windows provides functions that enable application programmers to use DNS, such as programmatic DNS query, record compare, and name lookup.

Programmable DNS components are designed for use by C/C++ programmers. Familiarity with networking and DNS is required. Programmers should be familiar with the IP-protocol suite, as well as the DNS protocol and DNS operations.

Run-time requirements

DNS is used on all IP networks that require an Internet-compatible locator service. However, the DNS API requires Windows 2000 or later.

In this section

TOPIC	DESCRIPTION
DNS Standards Documents	Links to public DNS standards documents.
About DNS	General information about DNS.
DNS Reference	Reference documentation for DNS.
DNS WMI Provider	General information and reference documentation for the DNS WMI provider.
Glossary	Glossary of DNS terms and definitions.

Related topics

DHCP

Internet Protocol Helper

HTTP Server API

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The HTTP Server API enables applications to communicate over HTTP without using Microsoft Internet Information Server (IIS). Applications can register to receive HTTP requests for particular URLs, receive HTTP requests, and send HTTP responses. The HTTP Server API includes SSL support so that applications can exchange data over secure HTTP connections without IIS. It is also designed to work with I/O completion ports.

Developer audience

This API is designed for use by C/C++ programmers.

Run-time requirements

The HTTP Server API is supported on Windows Server 2003 operating systems and on Windows XP with Service Pack 2 (SP2). Be aware that Microsoft IIS 5 running on Windows XP with SP2 is not able to share port 80 with other HTTP applications running simultaneously.

In this section

TOPIC	DESCRIPTION
About HTTP Server API	General information about HTTP.
Using HTTP Server API	Procedural guide for using HTTP.
HTTP Server API Reference	Documentation of specific functions, structures, and enumeration types available in HTTP.
HTTP Server Sample Application	A sample application that shows how to use the HTTP Server API to perform server-side tasks.

Related topics

Windows HTTP Services (WinHTTP)

IP Helper

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Internet Protocol Helper (IP Helper) API enables the retrieval and modification of network configuration settings for the local computer.

Where applicable

The IP Helper API is applicable in any computing environment where programmatically manipulating network and TCP/IP configuration is useful. Typical applications include IP routing protocols and Simple Network Management Protocol (SNMP) agents.

Developer audience

The IP Helper API is designed for use by C/C++ programmers. Programmers should also be familiar with Windows networking and TCP/IP networking concepts.

Run-time requirements

The IP Helper API can be used on all Windows platforms. Not all operating systems support all functions. Where certain implementations or capabilities of Windows Sockets 2 platform restrictions do exist, they are clearly noted in the documentation. If an IP Helper function is called on a platform that does not support the function, ERROR_NOT_SUPPORTED is returned. For more specific information about which operating systems support a particular function, refer to the Requirements sections in the documentation.

In this section

TOPIC	DESCRIPTION
What's New in IP Helper	Information on new features for IP Helper.
About IP Helper	General information about IP Helper programming considerations and capabilities available to developers.
Using IP Helper	Procedures and programming techniques used with IP Helper. This section includes basic IP Helper programming techniques, such as Getting Started With IP Helper.
IP Helper reference	Reference documentation for the IP Helper functions, structures, and enumerations.

Related topics

Windows Sockets 2

Routing and Remote Access Service

Network interfaces

2/18/2021 • 4 minutes to read • Edit Online

This topic describes high-level network interface concepts on Windows, including the ways they can be identified in code and their properties.

IMPORTANT

This topic is intended for a developer audience, both for Windows desktop networking apps and kernel mode networking drivers. Nevertheless, some of the information presented here can also be useful for system administrators managing network interfaces through PowerShell cmdlets.

Overview

A *network interface* is the point where two pieces of network equipment or protocol layers connect. Typically, this is represented by a physical Network Interface Card (NIC) for connection between a computer and a private or public network. However, it can also take the form of a software-only component such as the loopback interface (`127.0.0.1` for IPv4 or `::1` for IPv6).

Network interfaces are defined by the Internet Engineering Task Force (IETF) in RFC 2863 and are not meant to be defined by Windows. For detailed questions about the meaning of network interface identifiers such as **ifIndex**, see the IETF's definitions of them. The rest of this topic discusses Windows-specific implementation details.

Network interface identifiers and properties

On Windows, a network interface can be identified in different ways. Some of these identifiers are used to distinguish network interfaces from each other, but not all identifiers are equally suited to that task because of their differing properties. Generally, network interfaces are identified by a network address to external components. For example, this may be a node ID and a port number, or simply a unique node ID.

In code, a network interface can be identified in many ways. The following table details the ways a network interface can be identified along with associated properties. We recommend using the interface GUID (**ifGuid**) for programming unless a specific API requires a different network interface identifier.

NOTE

In the following table, **bolded** cells represent a property that is desirable for networking programmers.

IDENTIFIER	SIZE	IS UNIQUE ON THE SYSTEM	IS UNIQUE IN THE WORLD	IS PREDICTABLE	WILL BE RECYCLED IF THE NIC IS REMOVED	PERSISTS ACROSS REBOOTS	END USERS CAN MODIFY AT ANY TIME	DRIVERS CAN MODIFY AT ANY TIME	GENERAL FAMILIARITY WITH END USERS	IS ALWAYS PRESENT
ifIndex	**4 bytes**	**Yes**	No	No	Yes	No[1]	**No**	**No**	**Some[2]**	**Yes**

IDENTIFIER	SIZE	IS UNIQUE ON THE SYSTEM	IS UNIQUE IN THE WORLD	IS PREDICTABLE	WILL BE RECYCLED IF THE NIC IS REMOVED	PERSISTS ACROSS REBOOTS	END USERS CAN MODIFY AT ANY TIME	DRIVERS CAN MODIFY AT ANY TIME	GENERAL FAMILIARITY WITH END USERS	IS ALWAYS PRESENT
NetLuid	**8 bytes**	**Yes**	No	No	Yes	**Yes**	**No**	**No**	No	**Yes**
ifGuid	**16 bytes**	**Yes**	**Usually**[3]	No	**No**	**Yes**	**No**	**No**	No	**Yes**
ifAlias	514 bytes	**Yes for NICs**[4]	No	Sometimes[5]	Yes	**Yes**	Yes	**No**	**Yes**	**Usually**[4]
ifDescr	514 bytes	Usually[6]	No	No	Yes	**Yes**	**No**	Yes	**Yes**	**Usually**
ifPhysAddress (MAC ADDRESS)	0 to 32 bytes	Usually, for NICs	**Usually, for NICs**	**Yes**	**Tied to hardware**	**Yes**	**No**	**No**	**Yes**	**Usually**[7]
PnP instance ID	Up to 400 bytes	**Yes**	No	No	Yes	**Yes**	**No**	**No**	No	**Usually, for NICs**[8]
PnP location (PCI slot number)	Up to 400 bytes	**Yes**	No	**Yes**	Yes	**Yes**	**No**	**No**	Sometimes	Sometimes[8,9]

Notes for the preceding table:

1. **IfIndexes** are not guaranteed to be stable across reboots, even though they often receive the same value as the previous boot. Therefore, it is not recommended that drivers use **ifIndex** except where required by an API.
2. Some `netsh` commands take `ifIndex`, or `index`, as an input. Therefore, some administrative users are familiar with the **ifIndex** property if they use the `netsh` command frequently.
3. If a machine is cloned or imaged, then some of the GUIDs might be the same. Also, certain special network interfaces such as the built-in Teredo interface might have the same GUID on all machines.
4. NetCfg enforces that an **ifAlias** is a non-empty string and is unique among all NICs. However, the NDIS interface provider does not. Thererfore, it is possible to find special network interfaces with duplicate or empty names. This is most commonly seen with LBFO teams.
5. Only if the firmware supports Consistent Device Naming. Typcically, servers have this feature.
6. NetCfg assigns unique **ifDescrs** to all network interfaces. However, drivers can call an API to change the **ifDescr** to anything, including something that is not unique. Some 3rd party software packages do this.
7. Not all media types have a "MAC address." For example, some tunnels do not have this concept and simply advertise a zero-length byte array as their network address.

8. Only present on network interfaces that are backed by a PnP device. For example, loopback interfaces, light weight filter interfaces, interfaces provided by an NDIS interface provider, and certain special built-in NICs don't have PnP devices backing them.
9. Only some PnP buses support a PnP location ID. The built-in PCI and USB buses do, while root-enumerated devices do not.

Visibility to developers

In the preceding table, all properties except for the Plug and Play (PnP) properties are visible to user mode desktop apps and kernel mode drivers via a shared header (Netioapi.h). The PnP properties are visible via the Devpkey.h header and are used by both user mode desktop apps and kernel mode drivers. For example, see the DEVPKEY documentation.

The IP Helper API is also available for both user mode desktop apps and kernel mode drivers.

The UWP API surface only exposes the ifGuid property directly. However, it is possible for a UWP app developers to import the **GetIfTable2** function using P/Invoke if they are required to access other network interface properties.

Related topics

For management information base (MIB) definitions for network interfaces, see RFC 2863.

For NDIS network interfaces in network drivers, see NDIS Network Interfaces.

For Netioapi.h API reference, see netioapi.h header.

Network List Manager

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The Network List Manager API enables applications to retrieve a list of available network connections. Applications can filter networks, based on attributes and signatures, and choose the networks best suited to their task. The Network List Manager infrastructure notifies applications of changes in the network environment, thus enabling applications to dynamically update network connections.

Developer audience

Network List Manager is designed for use by COM programmers.

Run-time requirements

The Network List Manager API is supported starting with Windows Vista.

In this section

TOPIC	DESCRIPTION
About the Network List Manager API	General information about the Network List Manager API.
Network List Manager API Reference	Reference information for the Network List Manager API.

Network Management

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The network management functions provide the ability to manage user accounts and network resources. Many of the capabilities provided by the network management functions are not provided by other networking functions. However, if the capabilities are provided by another set of functions, the documentation for the network management functions will refer you to other functions you can use for the same task.

Developer audience

The network management functions are designed for use by C/C++ programmers. Programmers should also be familiar with Windows networking concepts

Run-time requirements

The network management functions can be used on all Windows platforms. Where certain implementations or capabilities of network management platform restrictions do exist, they are clearly noted in the documentation.

In this section

- What's New in Network Management
- About Network Management
- Using Network Management
- Network Management Reference

Network Share Management

2/18/2021 • 2 minutes to read • Edit Online

Network share management allows applications to manage and monitor communications between Windows clients and servers using the Server Messaging Block (SMB) protocol.

In this section

- About Network Share Management
- Network Share Management Reference

Peer-to-Peer

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Peer-to-peer technologies are used to facilitate real-time communication and collaboration across distributed networks.

In the peer-to-peer model, without using Internet servers, each PC user can do the following:

- Exchange data
- Share resources
- Locate other users
- Communicate
- Collaborate directly in real time

By using peer-to-peer technologies, applications that coordinate the use of computer CPU cycles and storage can share resources among small or large groups of computers connected to the Internet.

Where applicable

Developers can use the Peer Infrastructure to create a wide range of distributed, ad-hoc, and peer-to-peer applications.

Developer audience

Developers using the Peer Infrastructure should be familiar with C programming concepts. Developers using the PNRP Winsock Namespace Provider should be familiar with the Winsock API.

Run-time requirements

The Peer Infrastructure is supported in Windows Vista, Windows XP with Service Pack 2 (SP2) and later as well the Advanced Networking Pack for Windows XP available for Windows XP with Service Pack 1 (SP1). The Peer-to-Peer Infrastructure requires that IPv6 be installed and initiated to allow peer networking applications to function. Use of Peer-to-Peer Collaboration is only supported in Windows Vista .

In this section

TOPIC	DESCRIPTION
Peer Infrastructure	Information about the Peer Infrastructure and the Peer Name Resolution Protocol (PNRP).
Peer Collaboration	Information and reference material specific to the Peer Collaboration API.
Peer Distribution	Information and reference material specific to the Peer Distribution API.

Additional resources

Further information regarding Peer-to-Peer technologies can be found at the following locations:

Windows Peer Networking Resources	Access published white-papers, samples, and presentations detailing the Peer Networking technology.
Microsoft Peer Networking Blog	Read the latest blog entries from Microsoft's Peer Networking Team.
MSDN Peer Networking Forum	Discuss Peer technologies and collaborate with other developers.
TechNet Peer Networking Resources for IT Professionals	A conceptual Peer Networking overview, as well as guidance, specific to the IT Professional role.

Remote Procedure Call

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Microsoft Remote Procedure Call (RPC) defines a powerful technology for creating distributed client/server programs. The RPC run-time stubs and libraries manage most of the processes relating to network protocols and communication. This enables you to focus on the details of the application rather than the details of the network.

Where applicable

RPC can be used in all client/server applications based on Windows operating systems. It can also be used to create client and server programs for heterogeneous network environments that include such operating systems as Unix and Apple.

Developer audience

RPC is designed to be used by C/C++ programmers. Familiarity with the Microsoft Interface Definition Language (MIDL) and the MIDL compiler are required.

Run-time requirements

The RPC run-time libraries are included with Windows. The components of the RPC development environment are installed when you install the Microsoft Windows Software Development Kit (SDK). For details, see Installing the RPC Programming Environment.

In this section

TOPIC	DESCRIPTION
Best RPC Programming Practices	Guidance on RPC programming practices that help create the best possible RPC applications.
Overview	General information about incorporating RPC into your client/server applications.
Reference	Documentation of RPC types, functions, and constants.
RPC NDR Engine	Documentation of the marshaling engine for RPC and DCOM components, the RPC Network Data Representation (NDR) Engine.

Related topics

Microsoft Interface Definition Language (MIDL)

Routing and Remote Access Service

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Remote Access Service (RAS) can be used to create client applications. These applications display RAS common dialog boxes, manage remote access connections and devices, and manipulate phone-book entries.

The Routing APIs make it possible to create applications to administer the routing capabilities of the operating system.

Developers can use the Routing Protocol APIs to implement routing protocols.

Developer audience

The Routing and Remote Access Service APIs are designed for use by C/C++ programmers. Programmers should also be familiar with networking concepts.

Run-time requirements

For more specific information about which operating systems support a particular function, refer to the Requirements sections in the documentation.

In this section

TOPIC	DESCRIPTION
Routing and Remote Access Services Architecture	Overview of the Routing and Remote Access Services architecture.
Routing and Remote Access Registry Layout	An example registry layout for the router service
Routing and Remote Access Error Codes	List of all routing and remote access error codes.
Remote Access	Documentation for RAS and RAS Administration APIs.
Routing	Documentation for the Router Management and Management Information Base (MIB) APIs.
Routing Protocols	Documentation for the Multicast Group Manager, Routing Protocol Interface, and Routing Table Manager APIs.
Glossary	Definitions for terms used in the Routing and Remote Access Service documentation.

SNMP Service

11/2/2020 • 2 minutes to read • Edit Online

[SNMP is available for use in the operating systems specified in the Requirements section. It may be altered or unavailable in subsequent versions. Instead, use Windows Remote Management, which is the Microsoft implementation of WS-Man.]

Purpose

The Microsoft Windows implementation of the Simple Network Management Protocol (SNMP) is used to configure remote devices, monitor network performance, audit network usage, and detect network faults or inappropriate access.

IMPORTANT
The Microsoft Windows SNMP API only supports protocol versions up to SNMPv2C. It does not support any later versions of the protocol.

Where applicable

SNMP uses a distributed architecture consisting of management applications and agent applications. The SNMP service implements an SNMP agent. To use the information the SNMP service provides, you must have at least one host that is running an SNMP management application. You can use third-party SNMP management software, or you can develop your own SNMP management software application. The following APIs are available for this purpose:

- SNMP Management API, a set of functions that can be used to quickly develop basic SNMP management systems
- WinSNMP API, version 2.0, a set of functions for encoding, decoding, sending, and receiving SNMP messages

Additionally, the SNMP Extension Agent API defines the interface between the SNMP service and third-party SNMP extension agent DLLs. The SNMP Utility API functions can be used to simplify the processing of SNMP messages.

Developer audience

The APIs listed in the preceding section are designed for use by C/C++ programmers. Familiarity with SNMP and SNMPv2C, as well as a working knowledge of networking and network management concepts, are required.

Run-time requirements

For more information about the operating system required to use a particular function, see the Requirements section of the reference page for that function.

In this section

TOPIC	DESCRIPTION
New in SNMP	Information on updates to SNMP.

TOPIC	DESCRIPTION
Simple Network Management Protocol (SNMP)	Information and API reference for SNMP, including the SNMP Management API, SNMP Extension Agent API, and SNMP Utility API functions.
WinSNMP API	Information and API reference for the Microsoft Windows SNMP Application Programming Interface (WinSNMP API).

Related topics

Dynamic Host Configuration Protocol (DHCP)

Network Management

Routing and Remote Access Service

Telephony Application Programming Interfaces

2/18/2021 • 2 minutes to read • Edit Online

The Microsoft telephony application programming interfaces support the development of communications applications for Microsoft Windows. The telephony interfaces are listed in the following table.

INTERFACE	DESCRIPTION
TAPI 2.x	A C-programming language based API that enables you to implement communications applications ranging from basic modem control to call centers with multiple agents and switches.
TAPI 3.x	A COM-based API that merges classic and IP telephony.
TSPI	A telephony service provider (TSP) is a dynamic-link library (DLL) that supports communications device control through a set of exported service functions. A TAPI application uses standardized commands, TAPI passes information to the telephony service provider, and the TSP handles the specific commands that must be exchanged with the device.
MSPI	A media service provider (MSP) allows an application considerable control over the media for a particular transport mechanism. An MSP is always paired with a telephony service provider (TSP).

Teredo

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Teredo is an IPv6 transition technology that provides address assignment and host-to-host automatic tunneling for unicast IPv6 traffic when IPv6/IPv4 hosts are located behind one or multiple IPv4 network address translators (NATs). To traverse IPv4 NATs, IPv6 packets are sent as IPv4 User Datagram Protocol (UDP) messages.

Developer audience

Teredo is designed for use by C/C++ developers with IPv6 network programming experience.

Run-time requirements

The Teredo interface is primarily supported by Windows Vista and Windows Server 2008. The limited functionality of the Teredo Interface supported by Windows XP with Service Pack 2 (SP2) and Windows Server 2003 is detailed in Receiving Solicited Traffic Over Teredo.

In this section

TOPIC	DESCRIPTION
About Teredo	Information about the Teredo interface.
Using Teredo	Information about the implementation and general usage of the Teredo Interface.

WebSocket Protocol Component API

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The WebSocket Protocol Component API enables asynchronous, bi-directional communication channels over HTTP that work across existing network intermediaries. With the WebSocket Protocol Component API, a client uses HTTP to communicate with a server, and then both sides switch to using the underlying protocol that HTTP was layered on (such as TCP or SSL). The goal is to first use HTTP to traverse over network intermediaries, and then use the established end-to-end underlying TCP/SSL channel for bi-directional application communication. The WebSocket protocol [WSPROTO] is defined at the IETF, while an associated Javascript API [W3CAPI] is defined at the W3C.

In this section

TOPIC	DESCRIPTION
WebSocket Protocol Component API Data Types	The WebSocket Protocol Component API defines these data types.
WebSocket Protocol Component API Enumerations	The WebSocket Protocol Component API defines these enumerations.
WebSocket Protocol Component API Functions	The WebSocket Protocol Component API defines these functions.
WebSocket Protocol Component API Structures	The WebSocket Protocol Component API defines these structures.

Developer audience

The WebSocket Protocol Component API is designed for use by use by C/C++ programmers. Familiarity with HTTP and Windows networking is required.

NOTE
The preferred way to use the WebSocket protocol on Windows is through the Windows HTTP Services (WinHTTP) API or the Windows.Networking.Sockets namespace.

Run-time requirements

The WebSocket Protocol Component API requires Windows 8 and later versions of the Windows operating system. The APIs can be dynamically linked through websocket.dll.

NOTE
websocket.dll provides support for client and server handshake related HTTP headers, verifies received handshake data, and parses the WebSocket data stream. It does not handle any HTTP-specific operations (redirection, authentication, proxy support) nor perform any I/O operations (sending or receiving WebSocket stream bytes).

Related topics

HTTP

Windows HTTP Services (WinHTTP)

Windows Filtering Platform

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Windows Filtering Platform (WFP) is a set of API and system services that provide a platform for creating network filtering applications. The WFP API allows developers to write code that interacts with the packet processing that takes place at several layers in the networking stack of the operating system. Network data can be filtered and also modified before it reaches its destination.

By providing a simpler development platform, WFP is designed to replace previous packet filtering technologies such as Transport Driver Interface (TDI) filters, Network Driver Interface Specification (NDIS) filters, and Winsock Layered Service Providers (LSP). Starting in Windows Server 2008 and Windows Vista, the firewall hook and the filter hook drivers are not available; applications that were using these drivers should use WFP instead.

With the WFP API, developers can implement firewalls, intrusion detection systems, antivirus programs, network monitoring tools, and parental controls. WFP integrates with and provides support for firewall features such as authenticated communication and dynamic firewall configuration based on applications' use of sockets API (application-based policy). WFP also provides infrastructure for IPsec policy management, change notifications, network diagnostics, and stateful filtering.

Windows Filtering Platform is a development platform and not a firewall itself. The firewall application that is built into Windows Vista, Windows Server 2008, and later operating systems Windows Firewall with Advanced Security (WFAS) is implemented using WFP. Therefore, applications developed with the WFP API or the WFAS API use the common filtering arbitration logic that is built into WFP.

The WFP API consists of a user-mode API and a kernel-mode API. This section provides an overview of the entire WFP and describes in detail only the user-mode portion of the WFP API. For a detailed description of the kernel-mode WFP API, see the Windows Driver Kit online help.

Developer audience

The Windows Filtering Platform API is designed for use by programmers using C/C++ development software. Programmers should be familiar with networking concepts and design of systems using user-mode and kernel-mode components.

Run-time requirements

The Windows Filtering Platform is supported on clients running Windows Vista and later, and on servers running Windows Server 2008 and later. For information about the run-time requirements for a specific programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
What's New in Windows Filtering Platform	Information on new features and APIs in Windows Filtering Platform.
About Windows Filtering Platform	An overview of Windows Filtering Platform.

TOPIC	DESCRIPTION
Using Windows Filtering Platform	Example code using the Windows Filtering Platform API.
Windows Filtering Platform API Reference	Documentation for the Windows Filtering Platform functions, structures, and constants.

Additional resources

To ask questions and have discussions about using the WFP API, visit the Windows Filtering Platform Forum.

Related topics

Kernel-Mode Windows Filtering Platform API - Design Guide

Kernel-Mode Windows Filtering Platform API - Reference

Windows Firewall with Advanced Security

WFP Diagnostics Extensible Helper Class

Winsock Secure Socket Extensions

Windows Networking (WNet)

2/22/2020 • 2 minutes to read • Edit Online

The Windows networking (WNet) functions allow you to implement networking capabilities in your application without making allowances for a particular network provider or physical network implementation. This is because the WNet functions are network independent.

- About Windows Networking
- Using Windows Networking
- Windows Networking Reference

Windows Sockets 2

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Windows Sockets 2 (Winsock) enables programmers to create advanced Internet, intranet, and other network-capable applications to transmit application data across the wire, independent of the network protocol being used. With Winsock, programmers are provided access to advanced Microsoft® Windows® networking capabilities such as multicast and Quality of Service (QoS).

Winsock follows the Windows Open System Architecture (WOSA) model; it defines a standard service provider interface (SPI) between the application programming interface (API), with its exported functions and the protocol stacks. It uses the sockets paradigm that was first popularized by Berkeley Software Distribution (BSD) UNIX. It was later adapted for Windows in Windows Sockets 1.1, with which Windows Sockets 2 applications are backward compatible. Winsock programming previously centered around TCP/IP. Some programming practices that worked with TCP/IP do not work with every protocol. As a result, the Windows Sockets 2 API adds functions where necessary to handle several protocols.

Developer audience

Windows Sockets 2 is designed for use by C/C++ programmers. Familiarity with Windows networking is required.

Run-time requirements

Windows Sockets 2 can be used on all Windows platforms. Where certain implementations or capabilities of Windows Sockets 2 platform restrictions do exist, they are clearly noted in the documentation.

In this section

TOPIC	DESCRIPTION
What's New for Windows Sockets	Information on new features for Windows Sockets.
Winsock Network Protocol Support in Windows	Information on network protocol support for Windows Sockets on different versions of Windows.
About Winsock	General information on Windows Sockets programming considerations, architecture, and capabilities available to developers.
Using Winsock	Procedures and programming techniques used with Windows Sockets. This section includes basic Winsock programming techniques, such as Getting Started With Winsock, as well as advanced techniques useful for experienced Winsock developers.
Winsock Reference	Documentation of the Windows Sockets API.

Related topics

IP Helper

Quality of Service

Wireless Networking

2/18/2021 • 2 minutes to read • Edit Online

In this section

TOPIC	DESCRIPTION
Bluetooth	Bluetooth is an industry-standard protocol that enables wireless connectivity for computers, handheld devices, mobile phones, and other devices.
Infrared Data Association (IrDA)	Infrared Data Association (IrDA) is a protocol suite designed to provide wireless, line-of-sight connectivity between devices.
Mobile Broadband	The Mobile Broadband API is used to implement connectivity to cellular networks. Applications should not communicate with such mobile broadband devices directly. Instead, they must use the Mobile Broadband API.
Native Wifi	The Native Wifi automatic configuration component configures, connects to, and disconnects from wireless networks. Native Wifi can store profiles on the networks it interacts with in the form of XML documents.
Windows Connect Now	Windows Connect Now (WCN) allows mobile and embedded devices, 802.11 access points (APs), and computers to securely connect, and exchange settings with each other. WCN is designed for the home or small business user, providing a reasonable compromise between ease-of-use and robust security.
Windows Connection Manager	Windows Connection Manager (WCM) enables the creation and configuration of connection manager software.

Related topics

Internet

Networking

Bluetooth

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Bluetooth is an industry-standard protocol that enables wireless connectivity for computers, handheld devices, mobile phones, and other devices.

Where applicable

The Bluetooth application programming interface enables developers to use existing network programming knowledge to quickly develop or port applications.

Developer audience

Bluetooth is designed for use by C/C++ programmers. Some Bluetooth features are available with Windows Sockets. Familiarity with Microsoft Windows networking and Windows Sockets programming is required.

Run-time requirements

Microsoft Bluetooth support begins with Windows XP with Service Pack 1 (SP1).

Support for Bluetooth 2.1 is offered in Windows Vista SP2 and Windows 7.

In this section

TOPIC	DESCRIPTION
Overview	General information about Bluetooth.
Using	Information about using Bluetooth.
Reference	Reference documentation for Bluetooth.

Additional resources

Bluetooth Wireless Technology FAQ	Information about Bluetooth wireless technology support for Windows operating systems, focusing primarily on Windows Vista.
Windows Vista Wireless SDK Forum	Discuss Bluetooth implementation in relation to Windows Vista.
Windows XP Wireless SDK Forum	Discuss Bluetooth implementation in relation to Windows XP.

Mobile Broadband

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Mobile Broadband API is used to implement connectivity to cellular networks. Applications should not communicate with such mobile broadband devices directly. Instead, they must use the Mobile Broadband API.

Developer audience

The Mobile Broadband API is designed for C++ developers.

There is also an interface for .NET developers using C#. For details, read this whitepaper.

The Mobile Broadband API can be used by third-party applications that need to control and manage mobile broadband interfaces.

Run-time requirements

Mobile Broadband support begins with Windows 7.

In this section

- Mobile Broadband API Best Practices
- Mobile Broadband API Reference
- Mobile Broadband Profile Schema Reference

Native Wifi

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Native Wifi automatic configuration component configures, connects to, and disconnects from wireless networks. Native Wifi can store profiles on the networks it interacts with in the form of XML documents.

Developer audience

The Native Wifi API is designed for C/C++ developers. Programmers should be familiar with wireless networking concepts and terminology.

Run-time requirements

The Native Wifi component requires clients running Windows Vista, Windows XP with Service Pack 3 (SP3), or Wireless LAN API for Windows XP with Service Pack 2 (SP2).

In this section

TOPIC	DESCRIPTION
What's New in Native Wifi	Information on new features for Native Wifi.
About Native Wifi	General information on Native Wifi programming considerations, architecture, and capabilities available to developers.
Using Native Wifi	Procedures and programming techniques used with Native Wifi. This section includes basic Native Wifi programming techniques, such as Native Wifi API Sample, as well as more advanced techniques useful for experienced Native Wifi developers.
Native Wifi Reference	Documentation of the Native Wifi API and the XML profile schema.
Wireless Ad Hoc Reference	Documentation of the Native Wifi Ad Hoc interfaces and enumerations.
Wireless Zero Configuration Reference	Documentation of the client programming interface for the Wireless Zero Configuration module supported only on Windows XP .

Additional resources

For additional information on the Native Wifi API on older versions of Windows, visit the Windows Vista Wireless SDK Forum or the Windows XP Wireless SDK Forum.

Related topics

Wireless Networking

Windows Connect Now

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Windows Connect Now (WCN) allows mobile and embedded devices, 802.11 access points (APs), and computers to securely connect and exchange settings with each other. WCN is designed for the home or small business user, providing a reasonable compromise between ease-of-use and robust security.

Developer audience

The Windows Connect Now API is designed for use with C/C++.

Run-time requirements

Windows Connect Now is supported starting with Windows 7.

In this section

TOPIC	DESCRIPTION
About Windows Connect Now	A brief overview of the Windows Connect Now API.
Windows Connect Now Reference	Detailed descriptions of the programming elements that are included in the Windows Connect Now API (WCNAPI).

Windows Connection Manager

2/22/2020 • 2 minutes to read • Edit Online

Purpose

Windows Connection Manager (WCM) enables the creation and configuration of connection manager software.

Developer audience

Windows Connection Manager is designed for use by developers using C/C++ development software.

Run-time requirements

The Windows Connection Manager API is supported on Windows 8 and Windows Server 2012.

In this section

TOPIC	DESCRIPTION
Windows Connection Manager Reference	Reference documentation for the WCM API.

WebDAV

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Web Distributed Authoring and Versioning (WebDAV) is an extension to Hypertext Transfer Protocol (HTTP) that defines how basic file functions such as copy, move, delete, and create are performed by using HTTP. The WebDAV API is a set of Win32 functions for creating and managing connections to WebDAV servers and performing file I/O operations on remote files on WebDAV servers. WebDAV API functions are designed to be used together with other Win32 API functions such as Windows networking functions and file management functions.

Developer audience

The WebDAV API is designed for use by C and C++ developers.

Run-time requirements

The WebDAV API is in Windows Vista and later. For information about which operating system versions are required to use a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
WebDAV API Reference	Documentation of WebDAV functions.

Windows HTTP Services

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Microsoft Windows HTTP Services (WinHTTP) provides developers with an HTTP client application programming interface (API) to send requests through the HTTP protocol to other HTTP servers.

Where applicable

WinHTTP supports desktop client applications, Windows services, and Windows server-based applications.

For more information on how to use WinHTTP for applications built on the Microsoft .NET Framework, see the WinHttpHandler API

Developer audience

WinHTTP offers both a C/C++ application programming interface (API) and a Component Object Model (COM) automation component suitable for use in Active Server Pages (ASP) based applications.

A basic understanding of the HTTP protocol is important to use either interface.

Run-time requirements

WinHTTP 5.1 offers improvements over version 5.0. It is included in the operating system. For more information about new features, see What's New in WinHTTP 5.1 and What's New in Windows Server 2008 and Windows Vista.

IMPORTANT

The WinHTTP 5.0 download is no longer available. As of October 1, 2004, Microsoft removed the WinHTTP 5.0 SDK download from MSDN and terminated product support for version 5.0.

In this section

- About WinHTTP
- Using WinHTTP
- WinHTTP Reference

Windows Internet

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Microsoft Windows Internet (WinINet) application programming interface (API) enables applications to access standard Internet protocols, such as FTP and HTTP. For ease of use, WinINet abstracts these protocols into a high-level interface.

Where applicable

WinINet does not support server implementations. In addition, it should not be used from a service. For server implementations or services use Microsoft Windows HTTP Services (WinHTTP).

Developer audience

WinINet is designed for use by C/C++ programmers. It requires a basic understanding of the FTP and HTTP protocols.

Run-time requirements

Applications that use the WinINet API require Windows NT 4.0 or later, or Windows Me/98/95. For more information about which operating systems or components are required to use a particular programming element, see the Requirements section of the documentation.

In this section

TOPIC	DESCRIPTION
About Windows Internet	General information about the Windows Internet API.
Using Windows Internet	Programming guide for the Windows Internet API.
Windows Internet Reference	Reference documentation for the Windows Internet API.

Related topics

Microsoft Windows HTTP Services (WinHTTP)

Windows Web Services API

2/22/2020 • 2 minutes to read • Edit Online

Purpose

WWSAPI is a native-code implementation of SOAP which provides core network communication functionality by supporting a set of the WS-* and .NET-* family of protocols. WWSAPI is designed to be used by components/applications which fall into one of the following categories:

- Native code mandate
- Require minimal dependencies
- Require minimal startup time
- Memory constrained environments

Developer audience

Windows Web Services API (WWSAPI) offers C/C++ application programming interface (API) for building SOAP based web services and clients to them. A basic understanding of web services and protocol associated with them is important to use this API.

Run-time requirements

Windows Web Services API (WWSAPI) is an operating-system component of Windows 7 and Windows Server 2008 R2 or later versions of Microsoft Windows.

In this section

TOPIC	DESCRIPTION
About Windows Web Services	General information about the Windows Web Services API.
Using Windows Web Services	Programming guide for the Windows Web Services API.
Windows Web Services Reference	Reference documentation for the Windows Web Services API.

Security and Identity

2/18/2021 • 4 minutes to read • Edit Online

Develop more secure desktop apps by using Windows APIs and services. These APIs provide:

- Authentication
- Authorization
- Cryptography
- Directory, identity, and access services
- Parental controls
- Rights management

This section also provides best practices and other security articles.

In this section

TOPIC	DESCRIPTION
Antimalware Scan Interface	The Antimalware Scan Interface (AMSI) is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine. It provides enhanced malware protection for users and their data, applications, and workloads.
Authentication	Authentication is the process by which the system validates a user's logon information. A user's name and password are compared to an authorized list, and if the system detects a match, access is granted to the extent specified in the permission list for that user.
Authorization	Authorization is the right granted an individual to use the system and the data stored on it. Authorization is typically set up by a system administrator and verified by the computer based on some form of user identification, such as a code number or password.
Best Practices for the Security APIs	Provides best practices for developing more secure applications.
Certificate Enrollment API	The Certificate Enrollment API can be used to create a client application to request a certificate and install a certificate response.
Control Flow Guard (CFG)	Control Flow Guard (CFG) is a highly-optimized platform security feature that was created to combat memory corruption vulnerabilities.
Cryptography	Cryptography is the use of codes to convert data so that only a specific recipient will be able to read it, using a key. CryptoAPI enables users to create and exchange documents and other data in a secure environment, especially over nonsecure media such as the Internet.

TOPIC	DESCRIPTION
Cryptography API: Next Generation	Cryptography API: Next Generation (CNG) enable users to create and exchange documents and other data in a secure environment, especially over nonsecure media such as the Internet.
Dynamic Access Control developer extensibility	The Dynamic Access Control (DAC) scenario, as delivered in Windows Server 2012, has a variety of developer extensibility points that add customization potential for your applications development.
Directory, Identity, and Access Services	Network Administrators can use directory services to automate common administrative tasks, such as adding users and groups, managing printers, and setting permissions on network resources. Independent Software Vendors and end-user developers can use directory services to directory-enable their products and applications. Services can publish themselves in a directory, clients can use the directory to find services, and both can use the directory to find and manipulate other objects. Forefront Identity Manager (FIM) provides an integrated and comprehensive solution for managing the entire lifecycle of user identities and their associated credentials. Identity Lifecycle Manager (ILM) enables IT organizations to reduce the cost of managing the identity and access lifecycle by providing a single view of a user's identity across the heterogeneous enterprise and through the automation of common tasks. Active Directory Federation Service (AD FS) enables Federated Identity and Access Management by securely sharing digital identity and entitlements rights across security and enterprise boundaries.
Extensible Authentication Protocol	The Extensible Authentication Protocol (EAP) is a standard supported by several system components. EAP is crucial for protecting the security of wireless (802.1X) and wired LANs, Dial-up, and Virtual Private Networks (VPNs).
Extensible Authentication Protocol Host	EAPHost is a Microsoft Windows Networking component that provides an Extensible Authentication Protocol (EAP) infrastructure for the authentication of "supplicant" protocol implementations such as 802.1X and Point-to-Point (PPP).
MS-CHAP Password Management API	You can use the MS-CHAP Password Management API to create applications to change the passwords of networked users on remote workstations.
Network Access Protection	Network Access Protection (NAP) is a set of operating system components that provide a platform for protected access to private networks. The NAP platform provides an integrated way of evaluating the system health state of a network client that is attempting to connect to or communicate on a network and restricting the access of the network client until health policy requirements have been met.

TOPIC	DESCRIPTION
Network Policy Server	Network Policy Server (NPS) is the Microsoft implementation of a Remote Authentication Dial-in User Service (RADIUS) server and proxy. It is the successor of Internet Authentication Service (IAS).
Parental Controls	The Parental Controls technology in Windows is intended to assist diligent parents or guardians in ensuring access to appropriate materials by age or maturity level for those under their guardianship. It provides an extensible infrastructure in addition to built-in capabilities.
Rights Management	Three generations of Rights Management SDK are now available as well as an all-up roadmap to Microsoft supplied RMS code samples and developer tools across all supported operating systems; Android, iOS/OS X, Windows Phone and Windows Desktop.
Security Development Lifecycle (SDL) - Process Guidance	Microsoft Security Development Lifecycle (SDL) is an industry-leading software security assurance process. A Microsoft-wide initiative and a mandatory policy since 2004, the SDL has played a critical role in embedding security and privacy in Microsoft software and culture. Combining a holistic and practical approach, the SDL introduces security and privacy early and throughout all phases of the development process.
Security Management	The security management technologies can be used to manage Local Security Authority (LSA) policy and password filter policy, query the ability of programs from external sources, and service attachments that extend the functionality of the Security Configuration tool.
Security WMI Providers	The Security WMI providers enable administrators and programmers to configure BitLocker Drive Encryption (BDE) and the Trusted Platform Module (TPM) using Windows Management Instrumentation (WMI).
Security Glossary	Provides a glossary of security terms.
TPM Base Services	The Trusted Platform Module (TPM) Base Services (TBS) feature centralizes TPM access across applications. The TBS feature uses priorities specified by calling applications to cooperatively schedule TPM access.
Windows Biometric Framework API	You can use the Windows Biometric Framework API to create client applications that securely capture, save, and compare end-user biometric information.
Security Technical Articles	Articles on security and cryptography.

Antimalware Scan Interface (AMSI)

2/22/2020 • 2 minutes to read • Edit Online

Purpose

The Windows Antimalware Scan Interface (AMSI) is a versatile interface standard that allows your applications and services to integrate with any antimalware product that's present on a machine. AMSI provides enhanced malware protection for your end-users and their data, applications, and workloads.

AMSI is agnostic of antimalware vendor; it's designed to allow for the most common malware scanning and protection techniques provided by today's antimalware products that can be integrated into applications. It supports a calling structure allowing for file and memory or stream scanning, content source URL/IP reputation checks, and other techniques.

AMSI also supports the notion of a session so that antimalware vendors can correlate different scan requests. For instance, the different fragments of a malicious payload can be associated to reach a more informed decision, which would be much harder to reach just by looking at those fragments in isolation.

Windows components that integrate with AMSI

The AMSI feature is integrated into these components of Windows 10.

- User Account Control, or UAC (elevation of EXE, COM, MSI, or ActiveX installation)
- PowerShell (scripts, interactive use, and dynamic code evaluation)
- Windows Script Host (wscript.exe and cscript.exe)
- JavaScript and VBScript
- Office VBA macros

Developer audience, and sample code

The Antimalware Scan Interface is designed for use by two groups of developers.

- Application developers who want to make requests to antimalware products from within their apps.
- Third-party creators of antimalware products who want their products to offer the best features to applications.

For more info, see Developer audience, and sample code.

NOTE

Starting in Windows 10, version 1903, if your AMSI provider DLL is not Authenticode-signed, then it may not be loaded (depending on how the host machine is configured). For full details, see **IAntimalwareProvider** interface.

In this section

TOPIC	DESCRIPTION

TOPIC	DESCRIPTION
How AMSI helps you defend against malware	As an application developer, you can actively participate in malware defense. Specifically, you can help protect your customers from dynamic script-based malware, and from non-traditional avenues of cyberattack.
Developer audience, samples	This topic describes the groups of developers for whom the Antimalware Scan Interface is designed.
Antimalware Scan Interface Reference	Enumerations, COM interfaces, and other programming elements of the AMSI API.

Authentication (Authentication)

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Authentication is the process by which the system validates a user's logon information. A user's name and password are compared to an authorized list, and if the system detects a match, access is granted to the extent specified in the permission list for that user.

Microsoft authentication technologies include LSA Authentication, Credentials Management, Smart Card Authentication, Network Provider, Security Support Provider Interface (SSPI), Winlogon, and GINA.

Developer audience

Microsoft authentication technologies are intended for use by developers of applications that are based on the Windows Server, Windows Vista, and Windows operating systems that authenticate clients. Developers should be familiar with Windows-based programming. Although not required, an understanding of authentication or security-related subjects is advised.

Run-time requirements

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About Authentication	Key authentication concepts and a high-level view of Microsoft authentication technologies.
Using Authentication	Authentication processes, procedures, and examples of programs that use Microsoft authentication technologies.
Authentication Reference	Detailed information about authentication functions, interfaces, objects, structures, and other programming elements.

Authorization

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Authorization is the right granted an individual to use the system and the data stored on it. Authorization is typically set up by a system administrator and verified by the computer based on some form of user identification, such as a code number or password.

Microsoft authorization technologies include Authorization Manager and the Authz API.

Developer audience

Microsoft authorization technologies are intended for use by developers of applications based on the Windows Server and Windows operating systems that control access to resources. Developers should be familiar with Windows-based programming. Although not required, an understanding of authorization or security-related subjects is advised.

Run-time requirements

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About Authorization	Key authorization concepts and a high-level view of Microsoft authorization technologies.
Using Authorization	Authorization processes, procedures, and examples of programs using Microsoft authorization technologies. This information is presented in using sections for C++ and Script programming languages.
Authorization Reference	Detailed information about authorization functions, interfaces, structures, and other programming elements.

Best Practices for the Security APIs

2/18/2021 • 2 minutes to read • Edit Online

To help develop secure software, we recommend that you use the following best practices when developing applications. For more information, see Security Developer Center.

Security Development Life Cycle

The Security Development Life Cycle (SDL) is a process that aligns a series of security-focused activities and deliverables to each phase of software development. These activities and deliverables include:

- Developing threat models
- Using code-scanning tools
- Conducting code reviews and security testing

For more information about the SDL, see the SDL Blog.

Threat Models

Conducting a threat model analysis can help you discover potential points of attack in your code. For more information about threat model analysis, see Howard, Michael and LeBlanc, David [2003], *Writing Secure Code*, 2d ed., ISBN 0-7356-1722-8, Microsoft Press, Redmond, Washington. (This resource may not be available in some languages and countries.)

Service Packs and Security Updates

Build and test environments should mirror the same levels of service packs and security updates of the targeted user base. We recommend that you install the latest service packs and security updates for any Microsoft platform or application that is part of your build and test environment and encourage your users to do the same for the finished application environment. For more information about service packs and security updates, see Microsoft Windows Update and Microsoft Security.

Authorization

You should create applications that require the least possible privilege. Using the least possible privilege reduces the risk of malicious code compromising your computer system. For more information about running code in least possible privilege level, see Running with Special Privileges.

More Information

For more information about best practices, see the following topics.

TOPIC	DESCRIPTION
Running with Special Privileges	Discusses security implications of privileges.
Avoiding Buffer Overruns	Provides information about avoiding buffer overruns.
Control Flow Guard (CFG)	Discusses memory corruption vulnerabilities.

TOPIC	DESCRIPTION
Creating a DACL	Shows how to create a discretionary access control list (DACL) by using the Security Descriptor Definition Language (SDDL).
Handling Passwords	Discusses security implications of using passwords.
How to Optimize Your MSDN Library Search	Discusses options for searching Security SDK content on MSDN Library.
Dynamic Access Control developer extensibility	Basic orientation to some of the developer extensibility points for the new Dynamic Access Control solutions.

Certificate Enrollment API

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Certificate Enrollment API can be used to create a client application to request a certificate and install a certificate response. This API is implemented in CertEnroll.dll beginning with Windows Vista; it replaces Xenroll.dll.

Developer audience

The Certificate Enrollment API is for use by developers of applications that will enable users to create, request, and retrieve certificates over media, such as the Internet or an intranet, that are not inherently secure. Developers should be familiar with the C and C++ programming languages, the Component Object Model (COM), and the Windows-based programming environment. Although not required, an understanding of cryptography and public key infrastructure is advised.

Run-time requirements

The Certificate Enrollment API is supported beginning with Windows Server 2008 and Windows Vista. For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About the Certificate Enrollment API	Key concepts about certificate requests are discussed.
Using the Certificate Enrollment API	How to use the Certificate Enrollment API to extend the capabilities of Active Directory Certificate Services.
Certificate Enrollment API Reference	Detailed descriptions of interfaces, enumerations, and other programming elements that can be used to enroll a user or computer in a certificate hierarchy.

Cryptography

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Cryptography is the use of codes to convert data so that only a specific recipient will be able to read it, using a key.

Microsoft cryptographic technologies include CryptoAPI, Cryptographic Service Providers (CSP), CryptoAPI Tools, CAPICOM, WinTrust, issuing and managing certificates, and developing customizable public key infrastructures. Certificate and smart card enrollment, certificate management, and custom module development are also described.

Developer audience

CryptoAPI is intended for use by developers of Windows-based applications that will enable users to create and exchange documents and other data in a secure environment, especially over nonsecure media such as the Internet. Developers should be familiar with the C and C++ programming languages and the Windows programming environment. Although not required, an understanding of cryptography or security-related subjects is advised.

CAPICOM is a 32-bit only component that is intended for use by developers who are creating applications using Visual Basic Scripting Edition (VBScript) programming language or the C++ programming language. CAPICOM is available for use in the operating systems specified in Run-Time Requirements. For future development, we recommend that you use the .NET Framework to implement security features. For more information, see Alternatives to Using CAPICOM.

Run-time requirements

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

CAPICOM 2.1.0.2 is supported on the following operating systems and versions:

- Windows Server 2003
- Windows XP

CAPICOM is available as a redistributable file that can be downloaded from Platform SDK Redistributable: CAPICOM.

Certificate Services requires the following versions of these operating systems:

- Windows Server 2008 R2
- Windows Server 2008
- Windows Server 2003

In this section

TOPIC	DESCRIPTION

TOPIC	DESCRIPTION
About Cryptography	Key cryptography concepts and a high-level view of Microsoft cryptography technologies.
Using Cryptography	Cryptography processes, procedures, and extended samples of C and Visual Basic programs using CryptoAPI functions and CAPICOM objects.
Cryptography Reference	Detailed descriptions of the Microsoft cryptography functions, interfaces, objects, structures, and other programming elements. Includes reference descriptions of the API for working with digital certificates.

Cryptography API: Next Generation

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Cryptography API: Next Generation (CNG) is the long-term replacement for the *CryptoAPI*. CNG is designed to be extensible at many levels and cryptography agnostic in behavior.

Developer audience

CNG is intended for use by developers of applications that will enable users to create and exchange documents and other data in a secure environment, especially over nonsecure media such as the Internet. Developers should be familiar with the C and C++ programming languages and the Windows-based programming environment. Although not required, an understanding of cryptography or security-related subjects is advised.

If you are developing a CNG cryptographic algorithm provider or key storage provider, you must download the Cryptographic Provider Development Kit from Microsoft.

Run-time requirements

CNG is supported beginning with Windows Server 2008 and Windows Vista. For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About CNG	Describes CNG features, cryptographic primitives, and key storage, retrieval, import, and export.
Using CNG	Explains how to use the cryptography configuration features of CNG and typical CNG programming.
CNG Reference	Detailed descriptions of the CNG programming elements. These pages include reference descriptions of the API for working with CNG.

Active Directory Domain Services

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Microsoft Active Directory Domain Services are the foundation for distributed networks built on Windows 2000 Server, Windows Server 2003 and Microsoft Windows Server 2008 operating systems that use domain controllers. Active Directory Domain Services provide secure, structured, hierarchical data storage for objects in a network such as users, computers, printers, and services. Active Directory Domain Services provide support for locating and working with these objects.

This guide provides an overview of Active Directory Domain Services and sample code for basic tasks, such as searching for objects and reading properties, to more advanced tasks such as service publication.

Windows 2000 Server and later operating systems provide a user interface for users and administrators to work with the objects and data in Active Directory Domain Services. This guide describes how to extend and customize that user interface. It also describes how to extend Active Directory Domain Services by defining new object classes and attributes.

NOTE

The following documentation is for computer programmers. If you are an end-user trying to debug a printing error or home network issue, see the Microsoft community forums.

Where applicable

Network administrators write scripts and applications that access Active Directory Domain Services to automate common administrative tasks, such as adding users and groups, managing printers, and setting permissions for network resources.

Independent software vendors and end-user developers can use Active Directory Domain Services programming to directory-enable their products and applications. Services can publish themselves in Active Directory Domain Services; clients can use Active Directory Domain Services to find services, and both can use Active Directory Domain Services to locate and work with other objects on a network.

Developer audience

Applications that access data in Active Directory Domain Services can be written using the Active Directory Service Interfaces API, Lightweight Directory Access Protocol API, or the System.DirectoryServices namespace.

Run-time requirements

Active Directory Domain Services run on Windows 2000 and later domain controllers. However, client applications can be written for and run on Windows Vista, Windows Server 2003, Windows XP, Windows 2000, Windows NT 4.0, Windows 98, and Windows 95.

In this section

About Active Directory Domain Services

General information about Active Directory Domain Services.

Using Active Directory Domain Services

Active Directory Domain Services programming guide.

Active Directory Domain Services Reference

Active Directory Domain Services programming reference.

Active Directory Schema (AD Schema)

11/2/2020 • 2 minutes to read • Edit Online

The Microsoft Active Directory schema contains formal definitions of every object class that can be created in an Active Directory forest. The schema also contains formal definitions of every attribute that can exist in an Active Directory object. This section provides the reference for each schema object and provides a brief explanation of the attributes, classes, and other objects that make up the Active Directory schema.

NOTE

The following documentation contains the programming reference for Active Directory schema. If you are an end-user attempting to debug a printer error, try searching on the Microsoft community site. If you are a developer looking for a general overview of Active Directory schema, see the Active Directory Schema overview topics. If you are looking for programming guidelines for updating or modifying the schema, For more information about extending and customizing the schema, see Extending the Schema, as well as many of the topics in the Active Directory Domain Services and Active Directory Lightweight Directory Services programming guides.

In each of the reference topics, there is a section for each operating system that the topic applies to. The following operating systems are currently supported.

PLATFORM	NAME IN TOPIC
Microsoft Windows Server 2003	Windows Server 2003
Microsoft Active Directory Application Mode (ADAM)	ADAM
Microsoft Windows Server 2003 R2	Windows Server 2003 R2
Microsoft Windows Server 2008	Microsoft Windows Server 2008
Microsoft Windows Server 2008 R2	Microsoft Windows Server 2008 R2
Microsoft Windows Server 2012	Microsoft Windows Server 2012

If an operating system is not listed in the topic, the topic is not supported on that operating system. For example, if a topic only lists Windows Server 2003 and ADAM, then the topic does not apply to Windows Server 2003 R2.

The following sections contain detailed information about the Active Directory schema elements.

- Active Directory Schema Terminology
- Classes
- Attributes
- Syntaxes
- Control Access Rights
- RootDSE

Active Directory Service Interfaces

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Active Directory Service Interfaces (ADSI) is a set of COM interfaces used to access the features of directory services from different network providers. ADSI is used in a distributed computing environment to present a single set of directory service interfaces for managing network resources. Administrators and developers can use ADSI services to enumerate and manage the resources in a directory service, no matter which network environment contains the resource.

ADSI enables common administrative tasks, such as adding new users, managing printers, and locating resources in a distributed computing environment.

NOTE

The following documentation is for computer programmers. If you are an end-user trying to debug a printing error or home network issue, see the Microsoft community forums.

Where applicable

Network Administrators can use ADSI to automate common tasks, such as adding users and groups, managing printers, and setting permissions on network resources.

Independent Software Vendors and end-user developers can use ADSI to "directory enable" their products and applications. Services can publish themselves in a directory, clients can use the directory to find the services, and both can use the directory to find and manipulate other objects of interest. Because Active Directory Service Interfaces are independent of the underlying directory service(s), directory-enabled products and applications can operate successfully in multiple network and directory environments.

Developer audience

You can write ADSI client applications in many languages. For most administrative tasks, ADSI defines interfaces and objects accessible from Automation-compliant languages like Microsoft Visual Basic, Microsoft Visual Basic Scripting Edition (VBScript), and Java to the more performance and efficiency-conscious languages such as C and C++. A good foundation in COM programming is useful to the ADSI programmer.

Run-time requirements

Active Directory runs on Windows Server domain controllers. However, client applications using ADSI may be written and run on Windows. In addition, developers will want the Platform Software Development Kit (SDK), also available on the MSDN website. To investigate the contents of Active Directory, use the Active Directory Users and Computers MMC snap-in. This snap-in replaces the Adsvw tool that was available for previous versions of Windows.

In this section

About ADSI

General information about ADSI.

Using ADSI

Programmer's Guide to using ADSI.

ADSI Quick-start Tutorials

Using ADSI with Automation to manage directories.

ADSI Reference

Documentation of ADSI interfaces and methods.

Related topics

The Component Object Model

COM Clients and Servers

Active Directory Domain Services

Active Directory Federation Services

2/18/2021 • 2 minutes to read • Edit Online

Active Directory Federation Service (AD FS) is an authentication and authorization service that runs on Windows Server. AD FS provides users with single sign-on access between enterprise environments and web applications.

In this section

Active Directory Federation Services Overview

Overview information about AD FS.

ADFS Class Library

APIs used to develop application that use AD FS.

Related topics

Access Management Services

Extensible Authentication Protocol

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Extensible Authentication Protocol (EAP) is a standard supported by several system components. EAP is crucial for protecting the security of wireless (802.1X) and wired LANs, Dial-up, and Virtual Private Networks (VPNs).

Where applicable

EAP improves on previous authentication protocols such as Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP).

For new EAP method development, see Extensible Authentication Protocol Host.

Developer audience

The EAP API is designed for use by C/C++ programmers. Programmers should be familiar with networking concepts.

Run-time requirements

EAP is supported on client and server computers running on Windows 2000 and later. EAP is also supported on computers running on Windows 2000 Server and later if they are running Internet Authentication Service (IAS). For more information about supported operating systems, see the Requirements section in the documentation.

Related topics

Remote Access Service

Internet Authentication Service

Using Extensible Authentication Protocol

Extensible Authentication Protocol Reference

Extensible Authentication Protocol Host

2/18/2021 • 2 minutes to read • Edit Online

Purpose

EAPHost is a Microsoft Windows Networking component that provides an Extensible Authentication Protocol (EAP) infrastructure for the authentication of "supplicant" protocol implementations such as 802.1X and Point-to-Point (PPP). It also allows for authentication with "authenticator" technologies such as the Microsoft network policy server (NPS). Unlike the previous IAS Server (RADIUS), NPS supports Network Access Protection (NAP).

The EAPHost APIs enable applications to authenticate using the EAPHost service, and provide a template for the development of conformant authentication methods for use with EAPHost.

Run-time requirements

The EAPHost APIs are supported only in Windows Vista and later operating systems.

Related topics

About EAPHost

Using EAPHost

EAPHost API Reference

Network Access Protection

2/22/2020 • 2 minutes to read • Edit Online

Purpose

NOTE

The Network Access Protection platform is not available starting with Windows 10

Network Access Protection (NAP) is a set of operating system components that provide a platform for protected access to private networks. The NAP platform provides an integrated way of evaluating the system health state of a network client that is attempting to connect to or communicate on a network and restricting the access of the network client until health policy requirements have been met.

NAP is an extensible platform that provides an infrastructure and an API set for adding components that store, report, validate, and correct a computer's system health state. By itself, the NAP platform does not provide components to accumulate and evaluate attributes of a computer's health state. Other components, known as system health agents (SHAs) and system health validators (SHVs), provide network policy validation and network policy compliance.

Where applicable

NAP is designed to be extensible. It can interoperate with any vendor software that provides SHAs and SHVs or that recognizes its published API set. NAP helps provide a solution for the following common scenarios:

- Check the health and status of roaming laptops.
- Ensure the health of desktop computers.
- Verify the compliance and health of computers in remote offices.
- Determine the health of visiting laptops.
- Verify the compliance and health of unmanaged home computers.

Developer audience

The NAP API is designed for C/C++ developers. For the NAP enforcement methods, programmers should be familiar with networking protocols and technologies such as Remote Authentication Dial-in User Service (RADIUS), Dynamic Host Configuration Protocol (DHCP), virtual private networks (VPNs), the IEEE 802.1X standard for wired and wireless access, and Internet Protocol security (IPsec).

Run-time requirements

The NAP platform requires NAP infrastructure servers running Windows Server 2008 or later and NAP clients running Windows XP with Service Pack 3 (SP3), Windows Vista, or later operating systems. For specific information about which operating systems support a particular programming element, refer to the Requirements sections of the NAP APIs in the NAP Reference documentation.

In this section

TOPIC	DESCRIPTION
About NAP	General information about NAP API.
Using NAP	Usage examples for NAP API.
NAP Reference	Documentation for NAP interfaces, structures, and other code elements.

Network Policy Server

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Network Policy Server (NPS) is the Microsoft implementation of a Remote Authentication Dial-in User Service (RADIUS) server and proxy. It is the successor of Internet Authentication Service (IAS).

As a RADIUS server, NPS performs authentication, authorization, and accounting for wireless, authenticating switch, and remote access dial-up and virtual private network (VPN) connections.

NPS is also a health evaluator server for Network Access Protection (NAP). NPS performs authentication and authorization of network connection attempts and, based on configured system health policies, evaluates computer health compliance and determines how to limit a noncompliant computer's network access or communication. This is a new feature specific to NPS only; IAS does not support it. See Internet Authentication Service and Network Policy Server for a complete list of features new to NPS.

NPS includes two API sets: NPS Extensions API and Server Data Objects (SDO) API. Both NPS Extensions API and SDO API are also supported by the precursor of NPS, the Internet Authentication Service.

NPS Extensions API can be used to extend the authentication, authorization, and accounting methods offered by NPS and previously by IAS.

Server Data Objects API can be used to manipulate the network policy configuration on a computer that runs NPS or IAS.

NOTE

Network policies in NPS are equivalent to remote access policies in IAS.

Developer audience

The NPS Extensions API is designed for use by programmers using C/C++ development software. Programmers should be familiar with networking concepts and the RADIUS protocol. RADIUS is documented in RFC 2865 and RFC 2866.

The Server Data Objects API is designed for use by programmers using C/C++ or Visual Basic development software. Programmers should be familiar with Remote Access Service (RAS) and the RADIUS protocol.

Run-time requirements

NPS Extensions API is supported on Windows Server 2008 with the installation of the Microsoft Commercial Internet Service (MCIS).

Server Data Objects API is supported on Windows Server 2008.

NPS is available on Windows Server 2008 with the installation of the Microsoft Commercial Internet Service (MCIS).

In this section

Overview

General information regarding RADIUS, IAS, and NPS.

Network Policy Server Extensions API

API for implementing extension DLLs used for authentication, authorization, and accounting.

Server Data Objects API

API for managing the network policy configuration.

SQL Programmability

Samples of stored procedures used for managing NPS (IAS) logging.

Related topics

TechNet: Network Policy Server

TechNet: Internet Authentication Service

Network Access Protection

Parental Controls

2/18/2021 • 2 minutes to read • Edit Online

Purpose

As part of the Family Safety efforts of Microsoft, Windows provides Parental Controls functionality to monitor and limit exposure of selected computer users to online dangers and inappropriate content.

The Parental Controls technology in Windows is intended to assist diligent parents or guardians in ensuring access to appropriate materials by age or maturity level for those under their guardianship. It provides an extensible infrastructure in addition to built-in capabilities.

Developer audience

Independent software vendors (ISVs) of products that potentially expose users to safety risks or that are currently working to protect users from such risks are encouraged to integrate Parental Controls.

Run-time requirements

This functionality was introduced in Windows Vista. Parental Controls functionality is only available on selected consumer operating systems; it is not designed for use with versions of Windows Server. This functionality is not supported in Windows 10.

In this section

- What's New in Windows 8 Parental Controls
- What's New in Windows 7 Parental Controls
- About Parental Controls
- Using Parental Controls
- Parental Controls Reference
- Parental Controls Samples
- Parental Controls Glossary

Microsoft Rights Management SDKs

2/18/2021 • 2 minutes to read • Edit Online

Three generations of Rights Management SDK are now available: Rights Management SDK 4.2, Microsoft Rights Management SDK 2.1 and Active Directory Rights Management Services SDK, as well as a scripting API, Active Directory Rights Management Services Scripting API, that enables custom administration of an RMS server.

In this section

Code Samples and Tools

A collection of Microsoft supplied RMS code samples and developer support tools across all supported operating systems; Android, iOS/OS X, Windows Phone and Windows Desktop.

RMS SDK 4.2 is a simplified, next-generation tool set that provides a lightweight development experience in enabling your Android, iOS, and Mac OS X device apps with information protection via Microsoft Rights Management services.

RMS SDK 2.1 is a powerful SDK offering for Windows desktop application developers and server based solution providers to enable their products with rights management.

Active Directory Rights Management Services SDK

NOTE

AD RMS SDK leveraging functionality exposed by the client in Msdrm.dll is available for use in Windows Server 2012, Windows 8, Windows Server 2008 R2, Windows 7, Windows Server 2008, and Windows Vista. It may be altered or unavailable in subsequent versions. Instead, use Microsoft Rights Management Services SDK 2.1, which leverages functionality exposed by the client in Msipc.dll.

Active Directory Rights Management Services Scripting API

The Windows Active Directory Rights Management Services (AD RMS) Scripting API can be used to create scripts to administer an AD RMS installation.

RMS Scenarios, Code and Tools

2/18/2021 • 2 minutes to read • Edit Online

This roadmap provides Microsoft supplied code samples that show you how it's done and code snippets that you can use in your applications.

ITEM	OPERATING SYSTEM	SUPPORTING SDK VERSION	DESCRIPTION
Read PFILE protected PDF	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**Read PFILE protected PDF** is a simple code example on our RMS Developer's Corner blog that uses the MSIPC File API to decrypt and open a PFILE protected PDF document.
IpcManagedAPI	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcManagedAPI** is a .NET (C#) representation of RMS SDK 2.1 to make it easy for your managed application to be RMS-enabled.
IPCNotepad	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IPCNotepad** is a sample RMS-enabled application that takes you through the basic steps that each RMS-enabled application should perform when protecting and consuming restricted content.
IpcDlp	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcDlp** is a sample RMS-enabled Data Leak Protection (DLP) application that takes you through the basic steps that a DLP RMS-enabled application should perform by using File API for protecting and consuming restricted content.
IpcAzureApp	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcAzureApp** is a sample that demonstrates how to use RMS SDK in Azure application to protect data in Azure Blob Storage.
RmsDocumentInspector	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**RmsDocumentInspector** is a tool can give information about any RMS protected file such as content-id or user rights.

ITEM	OPERATING SYSTEM	SUPPORTING SDK VERSION	DESCRIPTION
RmsFileWatcher	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**RmsFileWatcher** is a sample that demonstrates how to build a Windows application that watches directories in the file system and applies RMS protection policies on every change, for example file added or file modified.
iOS/OS X usage scenarios	iOS / OS X	RMS SDK 4.2 and later versions of the 4.x SDK	**Objective C** code examples representing important development scenarios to get you accustomed to the RMS SDK. Examples include use of Microsoft Protected File format, custom protected file formats, and custom UI controls.
UI Library and Sample app	iOS	RMS SDK 4.2 and later versions of the 4.x SDK	**UI libraries and sample app for iOS** at GitHub, so you can get started quickly and re-use our standard UI in your apps.
UI Library and Sample app	Android	RMS SDK 4.2 and later versions of the 4.x SDK	**UI libraries and sample app for Android** at GitHub, so you can get started quickly and re-use our standard UI in your apps.
Android usage scenarios	Android	RMS SDK 4.2 and later versions of the 4.x SDK	**Java** code examples representing important development scenarios to get you accustomed to the RMS SDK. Examples include use of Microsoft Protected File format, custom protected file formats, and custom UI controls.

Security Management

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The security management technologies can be used to manage *Local Security Authority* (LSA) policy and *password filter* policy, query the ability of programs from external sources, and service attachments that extend the functionality of the Security Configuration tool.

Microsoft security management technologies include the LSA Policy API, the Password Filter API, the Safer API, and the Service Security Attachments API. These technologies enable software developers to create applications that manage systems and applications.

Developer audience

The LSA Policy API, the Password Filter API, the Safer API, and the Service Security Attachments API are intended for use by developers of applications that enable administrators to manage and secure their systems. Developers should be familiar with the C and C++ programming languages and with Windows-based programming.

Run-time requirements

For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About Security Management	An overview of the security management APIs.
Using Security Management	Common security management tasks.
Security Management Reference	Detailed information about the functions, structures, and other programming elements used to implement security management.

Security Glossary

2/18/2021 • 2 minutes to read • Edit Online

This section provides a glossary of security terms.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

TPM Base Services

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Trusted Platform Module (TPM) Base Services (TBS) feature centralizes TPM access across applications.

The TBS feature runs as a system service in Windows Server 2008, Windows Vista, or newer operating systems. It provides services as an API exposed through remote procedure calls (RPC). The TBS feature uses priorities specified by calling applications to cooperatively schedule TPM access.

NOTE

The TPM can be used for key storage operations. However, developers are encouraged to use the Key Storage APIs for these scenarios instead. The Key Storage APIs provide the functionality to create, sign or encrypt with, and persist cryptographic keys, and they are higher-level and easier to use than the TBS for these targeted scenarios.

Developer audience

TBS is intended for use by developers of applications based on the Windows operating systems. Developers should be familiar with the C and C++ programming languages and the Microsoft Windows programming environment.

Run-time requirements

The TBS feature requires at least Windows Server 2008 or Windows Vista operating system. For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
About TBS	Key concepts and a high-level view of the TBS feature.
Using TBS	TBS processes and procedures for using the TBS API.
TBS Reference	Documentation about the TBS functions, structures, and return codes.

Windows Biometric Framework

2/22/2020 • 2 minutes to read • Edit Online

Purpose

You can use the Windows Biometric Framework API to create client applications that securely capture, save, and compare end-user biometric information.

Developer audience

Developers who use this API should be familiar with the C and C++ programming languages and the Windows-based programming environment.

Run-time requirements

The Windows Biometric Framework API is supported beginning with Windows Server 2008 R2 and Windows 7. For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

TOPIC	DESCRIPTION
Biometric Framework overview	Describes the native support for biometrics in Windows.
Creating client applications	You can use the client API to capture and use biometric information in your applications.
Creating Adapter Plug-ins	You can create engine adapters, sensor adapters, and storage adapters using the topics in this section.
Creating a Private Sensor Pool	There are two types of sensor pools: private and system. This section describes each and explains how to create a private sensor pool.
Windows Biometric Framework API Reference	Detailed descriptions of functions, structures, and other programming elements that can be used to create a client applications and plug-ins.

System Services

2/18/2021 • 2 minutes to read • Edit Online

This section contains the reference for the system APIs and services offered by Windows for desktop apps. These include the traditionally available services for:

- The Component Object Model (COM).
- File compression.
- Dynamic-link libraries.
- Memory management.
- Power management.
- The creation and coordination of multiple threads of execution..
- The development of service applications.
- Windows messaging.
- Obtaining Windows system information.
- The Help API.

In this section

TOPIC	DESCRIPTION
COM	COM is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents) and ActiveX (Internet-enabled components) technologies.
COM+	COM+ is an evolution of Microsoft Component Object Model (COM) and Microsoft Transaction Server (MTS). COM+ builds on and extends applications written using COM, MTS, and other COM-based technologies. COM+ handles many of the resource management tasks that you previously had to program yourself, such as thread allocation and security. COM+ also makes your applications more scalable by providing thread pooling, object pooling, and just-in-time object activation. COM+ also helps protect the integrity of your data by providing transaction support, even if a transaction spans multiple databases over a network.
Compression API	The Compression API exposes the Windows MSZIP, XPRESS, XPRESS_HUFF, and LZMS compression algorithms. This enables developers of Windows applications to manage versions, service, and extend the exposed compression algorithms.
Distributed Transaction Coordinator	Guide and reference documentation for system administrators and developers using the Distributed Transaction Coordinator (DTC).
Microsoft.Dtc.PowerShell.Diagnostics	Provides information about the PowerShell cmdlets provided with Microsoft Distributed Transaction Coordinator (MSDTC) for diagnostics.

TOPIC	DESCRIPTION
Microsoft.MsDtcManagement.Commands	Provides information about the PowerShell cmdlets provided with Microsoft Distributed Transaction Coordinator (MSDTC) for management.
Dynamic Link Libraries	How to create and manage DLLs.
Help API	The Help API allows the opening of help catalogs and the retrieval of help content items.
Interprocess Communications	How to use mailslots and pipes.
Kernel Transaction Manager	How to use transacted file and registry operations, or define transactions for other resources.
Memory Management	Core memory management services.
MultiPoint Services	Server role that allows multiple users to simultaneously use the same computer, such as in a classroom environment.
Operation Recorder	Operation Recorder enables applications to speed up operations that repeatedly access the same file data by exposing the Windows prefetching mechanism as a public interface.
Power Management	Core power management services.
Processes and Threads	How to create and manage processes and threads.
Remote Desktop Services	How to programmatically interact with Remote Desktop Services.
Services	How to create and manage services.
Synchronization	How to coordinate multiple threads of execution.
Windows Desktop Sharing	Windows Desktop Sharing is a multiple-party screen-sharing technology. Key scenarios include remote assistance, real-time collaboration and conferencing, and video communication.
Windows Notification Framework	Documents the functions (and function callback prototypes) used to detect and possibly repair an application after a setup or migration has occurred.
Windows Subsystem for Linux	Reference information for the Windows Subsystem for Linux (WSL) programming interfaces.
Windows System Information	How to programmatically access the registry and key system configuration and version information.

Component Object Model (COM)

11/2/2020 • 2 minutes to read • Edit Online

Purpose

COM is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents) and ActiveX (Internet-enabled components) technologies.

Where applicable

COM objects can be created with a variety of programming languages. Object-oriented languages, such as C++, provide programming mechanisms that simplify the implementation of COM objects. These objects can be within a single process, in other processes, even on remote computers.

Run-time requirements

For information on which operating systems are required to use a particular interface or function, see the Requirements section of the documentation for the interface or function.

In this section

TOPIC	DESCRIPTION
COM Fundamentals	Describes the fundamental concepts and programming reference.
OLE and Data Transfer	Describes compound documents and data transfer.
Controls and Property Pages	Describes ActiveX controls and property pages.
COM Language Translations	Describes the differences between programming languages and describe how to translate COM object syntax from one language to another.

Related documentation

TOPIC	DESCRIPTION
COM Fundamentals	Describes the fundamental concepts and programming reference.
OLE and Data Transfer	Describes compound documents and data transfer.
Controls and Property Pages	Describes ActiveX controls and property pages.
COM Language Translations	Describes the differences between programming languages and describe how to translate COM object syntax from one language to another.

Related topics

TOPIC	DESCRIPTION
Component Object Model (COM)	COM is a platform-independent, distributed, object-oriented system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents) and ActiveX (Internet-enabled components) technologies.
Automation	Automation enables software packages to expose their unique features to scripting tools and other applications. Automation uses the Component Object Model (COM), but may be implemented independently from other OLE features, such as in-place activation.
Microsoft Interface Definition Language (MIDL)	The Microsoft Interface Definition Language (MIDL) defines interfaces between client and server programs. Microsoft includes the MIDL compiler with the Platform Software Development Kit (SDK) to enable developers to create the interface definition language (IDL) files and application configuration files (ACF) required for remote procedure call (RPC) interfaces and COM/DCOM interfaces. MIDL also supports the generation of type libraries for OLE Automation.
Structured Storage	Structured Storage provides file and data persistence in COM by handling a single file as a structured collection of objects known as storages and streams.
COM+	COM+ is an evolution of Microsoft Component Object Model (COM) and Microsoft Transaction Server (MTS). COM+ builds on and extends applications written using COM, MTS, and other COM-based technologies.

Microsoft Interface Definition Language

11/2/2020 • 2 minutes to read • Edit Online

Purpose

The Microsoft Interface Definition Language (MIDL) defines interfaces between client and server programs. Microsoft includes the MIDL compiler with the Platform Software Development Kit (SDK) to enable developers to create the interface definition language (IDL) files and application configuration files (ACF) required for remote procedure call (RPC) interfaces and COM/DCOM interfaces. MIDL also supports the generation of type libraries for OLE Automation.

Where applicable

MIDL can be used in all client/server applications based on Windows operating systems. It can also be used to create client and server programs for heterogeneous network environments that include such operating systems as Unix and Apple. Microsoft supports the Open Group (formerly known as the Open Software Foundation) DCE standard for RPC interoperability.

Developer audience

When using MIDL with RPC, familiarity with C/C++ programming and the RPC paradigm is required. When using MIDL with COM, familiarity with C++ programming and the RPC paradigm as it applies to COM is required, or alternatively, familiarity with OLE Automation model scripting and type libraries is required.

Run-time requirements

The appropriate run-time libraries for using MIDL are included with Windows. The MIDL compiler and the components of the RPC development environment are installed when you install the Windows SDK. For more information, see Using the MIDL Compiler and Installing the RPC Programming Environment.

In this section

TOPIC	DESCRIPTION
Overview	General information about MIDL and the MIDL compiler.
Using the MIDL Compiler	Information about using the MIDL compilter to generate RPC stubs.
Interface Definitions and Type Libraries	Documentation of RPC-specific interface definitions and type libraries.
MIDL Command-Line Reference	Documentation of the MIDL compiler command-line switches.
MIDL Language Reference	The MIDL compiler language reference.

Related topics

Structured Storage

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Structured Storage provides file and data persistence in COM by handling a single file as a structured collection of objects known as storages and streams.

The purpose of Structured Storage is to reduce the performance penalties and overhead associated with storing separate objects in a single file. Structured Storage provides a solution by defining how to handle a single file entity as a structured collection of two types of objects storages and streams through a standard implementation called Compound Files. This enables the user to interact with, and manage, a compound file as if it were a single file rather than a nested hierarchy of separate objects.

Where applicable

Structured Storage can be used on Microsoft COM-based operating systems.

Developer audience

The Structured Storage documentation is intended for experienced C and C++ programmers and COM-based system developers.

Structured Storage primarily supports C and C++ programming languages, however any COM-based technology will also support any programming language that utilizes interface pointers.

A solid understanding of COM technologies is prerequisite to the developmental use of Structured Storage.

Run-time requirements

For more information about which operating systems are required to use a particular API element, see the Requirements section of the documentation for the element.

In this section

TOPIC	DESCRIPTION
Overview	General information about Structured Storage.
Using Structured Storage	Using information for Structured Storage.
Reference	Documentation of Structured Storage specific interfaces, functions, structures, and enumerations.
Samples	Code examples written in C++. For more information, see Names in IStorage, Property Set Header, Section, Storing Property Sets, and Using Structured Storage.

Related topics

The Component Object Model

COM+ (Component Services)

2/18/2021 • 2 minutes to read • Edit Online

Purpose

COM+ is an evolution of Microsoft Component Object Model (COM) and Microsoft Transaction Server (MTS). COM+ builds on and extends applications written using COM, MTS, and other COM-based technologies. COM+ handles many of the resource management tasks that you previously had to program yourself, such as thread allocation and security. COM+ also makes your applications more scalable by providing thread pooling, object pooling, and just-in-time object activation. COM+ also helps protect the integrity of your data by providing transaction support, even if a transaction spans multiple databases over a network.

Where applicable

COM+ can be used to develop enterprise-wide, mission-critical, distributed applications for Windows.

If you are a system administrator, you will be installing, deploying, and configuring COM+ applications and their components. If you are an application programmer, you will be writing components and integrating them as applications. If you are a tools vendor, you will be developing or modifying tools to work in the COM+ environment.

Developer audience

COM+ is designed primarily for Microsoft Visual C++ and Microsoft Visual Basic developers.

Run-time requirements

COM+ version 1.5 is included in Windows starting with Windows XP and Windows Server 2003. COM+ version 1.0 is included in Windows 2000.

In this section

- What's New in COM+ 1.5
- COM+ Developers Overview
- COM+ Services
- COM+ General Tasks
- COM+ Reference

Related topics

COM

Compression API

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Compression API exposes the Windows MSZIP, XPRESS, XPRESS_HUFF, and LZMS compression algorithms. This enables developers of Windows applications to manage versions, service, and extend the exposed compression algorithms.

Developer audience

The Compression API is designed for use by professional C/C++ developers of Windows applications. The Compression API exposes the lossless compression algorithms used by Windows though a public interface and user-mode API. The Compression API is the recommended method for Windows developers to control the compression algorithms. This feature is 64-bit on 64-bit Windows and 32-bit on 32-bit Windows.

Run-time requirements

The Compression API is available beginning with the Windows 8 or Windows Server 2012.

In this section

- Using the Compression API
- Compression API Functions
- Compression API Structures
- Compression API Enumerations

Dynamic-Link Libraries (Dynamic-Link Libraries)

2/18/2021 • 2 minutes to read • Edit Online

A *dynamic-link library* (DLL) is a module that contains functions and data that can be used by another module (application or DLL).

A DLL can define two kinds of functions: exported and internal. The exported functions are intended to be called by other modules, as well as from within the DLL where they are defined. Internal functions are typically intended to be called only from within the DLL where they are defined. Although a DLL can export data, its data is generally used only by its functions. However, there is nothing to prevent another module from reading or writing that address.

DLLs provide a way to modularize applications so that their functionality can be updated and reused more easily. DLLs also help reduce memory overhead when several applications use the same functionality at the same time, because although each application receives its own copy of the DLL data, the applications share the DLL code.

The Windows application programming interface (API) is implemented as a set of DLLs, so any process that uses the Windows API uses dynamic linking.

- About Dynamic-Link Libraries
- Using Dynamic-Link Libraries
- Dynamic-Link Library Reference

NOTE

If you are a user experiencing difficulty with a DLL on your computer, you should contact customer support for the software vendor that publishes the DLL. If you feel you are in need of support for a Microsoft product (including Windows), please go to our technical support site at support.microsoft.com.

Related topics

DLLs (Visual C++)

Interprocess Communications

2/18/2021 • 10 minutes to read • Edit Online

The Windows operating system provides mechanisms for facilitating communications and data sharing between applications. Collectively, the activities enabled by these mechanisms are called *interprocess communications* (IPC). Some forms of IPC facilitate the division of labor among several specialized processes. Other forms of IPC facilitate the division of labor among computers on a network.

Typically, applications can use IPC categorized as clients or servers. A *client* is an application or a process that requests a service from some other application or process. A *server* is an application or a process that responds to a client request. Many applications act as both a client and a server, depending on the situation. For example, a word processing application might act as a client in requesting a summary table of manufacturing costs from a spreadsheet application acting as a server. The spreadsheet application, in turn, might act as a client in requesting the latest inventory levels from an automated inventory control application.

After you decide that your application would benefit from IPC, you must decide which of the available IPC methods to use. It is likely that an application will use several IPC mechanisms. The answers to these questions determine whether an application can benefit by using one or more IPC mechanisms.

- Should the application be able to communicate with other applications running on other computers on a network, or is it sufficient for the application to communicate only with applications on the local computer?
- Should the application be able to communicate with applications running on other computers that may be running under different operating systems (such as 16-bit Windows or UNIX)?
- Should the user of the application have to choose the other applications with which the application communicates, or can the application implicitly find its cooperating partners?
- Should the application communicate with many different applications in a general way, such as allowing cut-and-paste operations with any other application, or should its communications requirements be limited to a restricted set of interactions with specific other applications?
- Is performance a critical aspect of the application? All IPC mechanisms include some amount of overhead.
- Should the application be a GUI application or a console application? Some IPC mechanisms require a GUI application.

The following IPC mechanisms are supported by Windows:

- Clipboard
- COM
- Data Copy
- DDE
- File Mapping
- Mailslots
- Pipes
- RPC
- Windows Sockets

Using the Clipboard for IPC

The clipboard acts as a central depository for data sharing among applications. When a user performs a cut or copy operation in an application, the application puts the selected data on the clipboard in one or more standard or application-defined formats. Any other application can then retrieve the data from the clipboard, choosing from the available formats that it understands. The clipboard is a very loosely coupled exchange medium, where applications need only agree on the data format. The applications can reside on the same computer or on different computers on a network.

Key Point: All applications should support the clipboard for those data formats that they understand. For example, a text editor or word processor should at least be able to produce and accept clipboard data in pure text format. For more information, see Clipboard.

Using COM for IPC

Applications that use OLE manage *compound documents*—that is, documents made up of data from a variety of different applications. OLE provides services that make it easy for applications to call on other applications for data editing. For example, a word processor that uses OLE could embed a graph from a spreadsheet. The user could start the spreadsheet automatically from within the word processor by choosing the embedded chart for editing. OLE takes care of starting the spreadsheet and presenting the graph for editing. When the user quit the spreadsheet, the graph would be updated in the original word processor document. The spreadsheet appears to be an extension of the word processor.

The foundation of OLE is the Component Object Model (COM). A software component that uses COM can communicate with a wide variety of other components, even those that have not yet been written. The components interact as objects and clients. Distributed COM extends the COM programming model so that it works across a network.

Key Point: OLE supports compound documents and enables an application to include embedded or linked data that, when chosen, automatically starts another application for data editing. This enables the application to be extended by any other application that uses OLE. COM objects provide access to an object's data through one or more sets of related functions, known as *interfaces*. For more information, see COM and ActiveX Object Services.

Using Data Copy for IPC

Data copy enables an application to send information to another application using the **WM_COPYDATA** message. This method requires cooperation between the sending application and the receiving application. The receiving application must know the format of the information and be able to identify the sender. The sending application cannot modify the memory referenced by any pointers.

Key Point: Data copy can be used to quickly send information to another application using Windows messaging. For more information, see Data Copy.

Using DDE for IPC

DDE is a protocol that enables applications to exchange data in a variety of formats. Applications can use DDE for one-time data exchanges or for ongoing exchanges in which the applications update one another as new data becomes available.

The data formats used by DDE are the same as those used by the clipboard. DDE can be thought of as an extension of the clipboard mechanism. The clipboard is almost always used for a one-time response to a user command, such as choosing the Paste command from a menu. DDE is also usually initiated by a user command, but it often continues to function without further user interaction. You can also define custom DDE data formats for special-purpose IPC between applications with more tightly coupled communications requirements.

DDE exchanges can occur between applications running on the same computer or on different computers on a network.

Key Point: DDE is not as efficient as newer technologies. However, you can still use DDE if other IPC mechanisms are not suitable or if you must interface with an existing application that only supports DDE. For more information, see Dynamic Data Exchange and Dynamic Data Exchange Management Library.

Using a File Mapping for IPC

File mapping enables a process to treat the contents of a file as if they were a block of memory in the process's address space. The process can use simple pointer operations to examine and modify the contents of the file. When two or more processes access the same file mapping, each process receives a pointer to memory in its own address space that it can use to read or modify the contents of the file. The processes must use a synchronization object, such as a semaphore, to prevent data corruption in a multitasking environment.

You can use a special case of file mapping to provide *named shared memory* between processes. If you specify the system swapping file when creating a file-mapping object, the file-mapping object is treated as a shared memory block. Other processes can access the same block of memory by opening the same file-mapping object.

File mapping is quite efficient and also provides operating-system–supported security attributes that can help prevent unauthorized data corruption. File mapping can be used only between processes on a local computer; it cannot be used over a network.

Key Point: File mapping is an efficient way for two or more processes on the same computer to share data, but you must provide synchronization between the processes. For more information, see File Mapping and Synchronization.

Using a Mailslot for IPC

Mailslots provide one-way communication. Any process that creates a mailslot is a *mailslot server*. Other processes, called *mailslot clients*, send messages to the mailslot server by writing a message to its mailslot. Incoming messages are always appended to the mailslot. The mailslot saves the messages until the mailslot server has read them. A process can be both a mailslot server and a mailslot client, so two-way communication is possible using multiple mailslots.

A mailslot client can send a message to a mailslot on its local computer, to a mailslot on another computer, or to all mailslots with the same name on all computers in a specified network domain. Messages broadcast to all mailslots on a domain can be no longer than 400 bytes, whereas messages sent to a single mailslot are limited only by the maximum message size specified by the mailslot server when it created the mailslot.

Key Point: Mailslots offer an easy way for applications to send and receive short messages. They also provide the ability to broadcast messages across all computers in a network domain. For more information, see Mailslots.

Using Pipes for IPC

There are two types of pipes for two-way communication: anonymous pipes and named pipes. *Anonymous pipes* enable related processes to transfer information to each other. Typically, an anonymous pipe is used for redirecting the standard input or output of a child process so that it can exchange data with its parent process. To exchange data in both directions (duplex operation), you must create two anonymous pipes. The parent process writes data to one pipe using its write handle, while the child process reads the data from that pipe using its read handle. Similarly, the child process writes data to the other pipe and the parent process reads from it. Anonymous pipes cannot be used over a network, nor can they be used between unrelated processes.

Named pipes are used to transfer data between processes that are not related processes and between processes on different computers. Typically, a named-pipe server process creates a named pipe with a well-known name or a name that is to be communicated to its clients. A named-pipe client process that knows the name of the pipe can open its other end, subject to access restrictions specified by named-pipe server process. After both the server and client have connected to the pipe, they can exchange data by performing read and write operations on the pipe.

Key Point: Anonymous pipes provide an efficient way to redirect standard input or output to child processes on the same computer. Named pipes provide a simple programming interface for transferring data between two processes, whether they reside on the same computer or over a network. For more information, see Pipes.

Using RPC for IPC

RPC enables applications to call functions remotely. Therefore, RPC makes IPC as easy as calling a function. RPC operates between processes on a single computer or on different computers on a network.

The RPC provided by Windows is compliant with the Open Software Foundation (OSF) Distributed Computing Environment (DCE). This means that applications that use RPC are able to communicate with applications running with other operating systems that support DCE. RPC automatically supports data conversion to account for different hardware architectures and for byte-ordering between dissimilar environments.

RPC clients and servers are tightly coupled but still maintain high performance. The system makes extensive use of RPC to facilitate a client/server relationship between different parts of the operating system.

Key Point: RPC is a function-level interface, with support for automatic data conversion and for communications with other operating systems. Using RPC, you can create high-performance, tightly coupled distributed applications. For more information, see Microsoft RPC Components.

Using Windows Sockets for IPC

Windows Sockets is a protocol-independent interface. It takes advantage of the communication capabilities of the underlying protocols. In Windows Sockets 2, a socket handle can optionally be used as a file handle with the standard file I/O functions.

Windows Sockets are based on the sockets first popularized by Berkeley Software Distribution (BSD). An application that uses Windows Sockets can communicate with other socket implementation on other types of systems. However, not all transport service providers support all available options.

Key Point: Windows Sockets is a protocol-independent interface capable of supporting current and emerging networking capabilities. For more information, see Windows Sockets 2.

Kernel Transaction Manager

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Kernel Transaction Manager (KTM) enables the development of applications that use transactions. The transaction engine itself is within the kernel, but transactions can be developed for kernel- or user-mode transactions, and within a single host or among distributed hosts.

The KTM is used to implement Transactional NTFS (TxF) and Transactional Registry (TxR). TxF allows transacted file system operations within the NTFS file system. TxR allows transacted registry operations. KTM enables client applications to coordinate file system and registry operations with a transaction.

To develop an application that coordinates transactions with resources other than TxF or TxR, you must first develop a Win32 transaction-aware service, also called a resource manager.

Managed and COM+ applications should use their native transaction managers.

Where applicable

KTM can be used with applications and resource managers hosted on Windows Vista or Windows Server 2008.

Developer audience

The KTM API is designed for use by C and C++ programmers.

Run-time requirements

KTM is supported starting with Windows Vista.

In this section

TOPIC	DESCRIPTION
About	General information about transactions and the capabilities provided by KTM.
Reference	Documentation for the functions, data structures, enumerations, and other programming elements of KTM.

Related topics

Common Log File System

Transactional NTFS (TxF)

Distributed Transaction Coordinator

Memory Management (Memory Management)

2/18/2021 • 2 minutes to read • Edit Online

The memory manager implements virtual memory, provides a core set of services such as memory mapped files, copy-on-write memory, large memory support, and underlying support for the cache manager.

In This Section

- About Memory Management
- Using the Memory Management Functions
- Memory Management Reference

Power Management

2/18/2021 • 2 minutes to read • Edit Online

Purpose

This overview describes the power management support in the Windows operating system and explains how to use it in your applications.

In this section

- About Power Management
- Using Power Management
- Power Management Reference

Processes and Threads

2/18/2021 • 2 minutes to read • Edit Online

An application consists of one or more processes. A *process*, in the simplest terms, is an executing program. One or more threads run in the context of the process. A *thread* is the basic unit to which the operating system allocates processor time. A thread can execute any part of the process code, including parts currently being executed by another thread.

A *job object* allows groups of processes to be managed as a unit. Job objects are namable, securable, sharable objects that control attributes of the processes associated with them. Operations performed on the job object affect all processes associated with the job object.

A *thread pool* is a collection of worker threads that efficiently execute asynchronous callbacks on behalf of the application. The thread pool is primarily used to reduce the number of application threads and provide management of the worker threads.

A *fiber* is a unit of execution that must be manually scheduled by the application. Fibers run in the context of the threads that schedule them.

User-mode scheduling (UMS) is a lightweight mechanism that applications can use to schedule their own threads. UMS threads differ from fibers in that each UMS thread has its own thread context instead of sharing the thread context of a single thread.

- What's New in Processes and Threads
- About Processes and Threads
- Using Processes and Threads
- Process and Thread Reference

Remote Desktop Services (Remote Desktop Services)

2/18/2021 • 3 minutes to read • Edit Online

Purpose

Windows Server 2012 R2, Windows Server 2012, Windows Server 2008 R2, or Windows Server 2008 with Remote Desktop Services (formerly known as Terminal Services) allow a server to host multiple, simultaneous client sessions. Remote Desktop uses Remote Desktop Services technology to allow a single session to run remotely. A user can connect to a Remote Desktop Session Host (RD Session Host) server (formerly known as a terminal server) by using Remote Desktop Connection (RDC) client software. The Remote Desktop Web Connection extends Remote Desktop Services technology to the web.

NOTE

This topic is for software developers. If you are looking for user information for Remote Desktop connections, See Remote Desktop Connection: frequently asked questions.

Where applicable

A Remote Desktop Connection (RDC) client can exist in a variety of forms. Thin-client hardware devices that run an embedded Windows-based operating system can run the RDC client software to connect to an RD Session Host server. Windows-, Macintosh-, or UNIX-based computers can run RDC client software to connect to an RD Session Host server to display Windows-based applications. This combination of RDC clients provides access to Windows-based applications from virtually any operating system.

Developer audience

Developers who use Remote Desktop Services should be familiar with the C and C++ programming languages and the Windows-based programming environment. Familiarity with client/server architecture is required. The Remote Desktop Web Connection includes scriptable interfaces to create and deploy scriptable virtual channels within Remote Desktop Services web applications.

Run-time requirements

Applications that use Remote Desktop Services require Windows Server 2012 R2, Windows 8.1, Windows Server 2012, Windows 8, Windows Server 2008 R2, Windows 7, Windows Server 2008, or Windows Vista. To use Remote Desktop Web Connection functionality, the Remote Desktop Services client application requires Internet Explorer and a connection to the World Wide Web. For information about run-time requirements for a particular programming element, see the Requirements section of the reference page for that element.

In this section

Remote Desktop ActiveX control

Describes how to use the Remote Desktop ActiveX control.

Remote Desktop Protocol Provider API

You use the Remote Desktop Protocol Provider API to create a protocol to provide communication between the Remote Desktop Services service and multiple clients.

Remote Desktop Services virtual channels

Virtual channels are software extensions that can be used to add functional enhancements to a Remote Desktop Services application.

RemoteFX Media Redirection API

The RemoteFX Media Redirection API is used in a Remote Desktop session to identify areas of the server that are displaying fast changing content, such as video. This content can then be video encoded and sent to the client in encoded format.

Remote Desktop Connection Broker client API

Describes how to use the Remote Desktop Connection Broker client API.

Personal desktop task agent API reference

The personal desktop task agent API is used to handle scheduled updates to a personal virtual desktop.

About Remote Desktop Services

Remote Desktop Services (formerly known as Terminal Services) provides functionality similar to a terminal-based, centralized host, or mainframe, environment in which multiple terminals connect to a host computer.

Remote Desktop Management Services Provider

The Remote Desktop Management Services (RDMS) Provider manages virtual desktop infrastructure (VDI) environments.

Remote Desktop Services reference

Documentation of property methods that you can use to examine and configure Remote Desktop Services user properties. Remote Desktop Services functions, structures, and Remote Desktop Web Connection scriptable interfaces are also documented.

Remote Desktop Services Shortcut Keys

A list of the Remote Desktop Services shortcut keys.

Remote Desktop Services WMI provider

The Remote Desktop Services WMI provider provides programmatic access to the information and settings that are exposed by the Remote Desktop Services Configuration/Connections Microsoft Management Console (MMC) snap-in.

Using Remote Desktop Services

How to program in the Remote Desktop Services environment and how to extend Remote Desktop Services (formerly known as Terminal Services) technology to the web by using Remote Desktop Web Connection.

Related topics

Terminal Services Is Now Remote Desktop Services

Services (Services)

2/18/2021 • 2 minutes to read • Edit Online

A *service application* conforms to the interface rules of the Service Control Manager (SCM). It can be started automatically at system boot, by a user through the Services control panel applet, or by an application that uses the service functions. Services can execute even when no user is logged on to the system.

A *driver service* conforms to the device driver protocols. It is similar to a service application, but it does not interact with the SCM. For simplicity, the term *service* refers to a *service application* in this overview.

Triggers can now be used to control service start. For more info, see Service Configuration.

- What's New in Services for Windows 8
- What's New in Services for Windows 7
- Service Changes for Windows Vista
- About Services
- Using Services
- Service Reference

Setup API

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Setup API provides a set of functions that a setup application calls to perform installation operations.

Where applicable

These setup functions are available to develop a setup application. Setup API should no longer be used for installing applications. Instead, use the Windows Installerfor developing application installers. Setup API continues to be used for installing device drivers.

The Setup API is intended for the development of desktop style applications.

Developer audience

A developer can use the Setup API if their setup application requires the following functionality:

- Queuing of files.
- Installation of files.
- Handling of installation errors and prompting for media.
- Updating registry entries.
- Logging of files as they are installed.
- Storage of the most recently used source paths.

Run-time requirements

For information about which operating systems are required to use a particular function, see the Requirements section of the documentation for the function.

In this section

TOPIC	DESCRIPTION
Overview	General information about Setup API.
Reference	Documentation of Setup API data types, structures, functions, and notifications.

Synchronization

2/18/2021 • 2 minutes to read • Edit Online

There are a variety of ways to coordinate multiple threads of execution. The functions described in this overview provide mechanisms that threads can use to synchronize access to a resource.

- What's New in Synchronization
- About Synchronization
- Using Synchronization
- Synchronization Reference

Windows System Information

2/18/2021 • 2 minutes to read • Edit Online

The following overviews describe the types of system information available.

OVERVIEW	DESCRIPTION
Handles and Objects	An *object* is a data structure that represents a system resource, such as a file, thread, or graphic image. An application cannot directly access object data or the system resource that an object represents. Instead, an application must obtain an object *handle*, which it can use to examine or modify the system resource.
Registry	A system-defined database in which applications and the system store and retrieve configuration information.
System Information	Retrieves or sets system configuration, settings, version, and metrics.
Time	Retrieves or sets the system time.
Time Provider	Retrieves accurate time stamps from hardware or the network, and provides time stamps to other clients on the network.
WaaS Assessment Platform	The Windows as a Service (WaaS) Update Assessment Platform provides information on a device's Windows updates.

Additional Resources

General Windows Development Issues Forum	Discuss Windows System Information and other general issues about developing applications for Windows.

Windows Machine Learning

2/18/2021 • 2 minutes to read • Edit Online

The following topics describe the COM reference pages for the Windows Machine Learning SDK.

In this section

TOPIC	DESCRIPTION
Windows Machine Learning Reference	The following topics describe the COM reference pages for the Windows Machine Learning SDK.

Developer Notes

2/18/2021 • 2 minutes to read • Edit Online

This documentation set describes items that may be of interest to developers. It is divided into the following sections:

- Application Compatibility Database
- Application Verifier
- Audio and Video
- Cabinet API
- Debugging
- Drivers
- Edition Upgrade API
- Feature Staging
- FeedbackHub
- File History API
- Files
- IME Share
- Graphics Low-Level Client Support
- Miscellaneous Low-Level Client Support
- MSHTML
- Installation
- iSCSI Target Pass-Through
- Java VM
- Jet Debugging
- MSN
- MSXML 1.0 (Obsolete)
- Networking
- NFS
- Offline Files
- Offline Registry Library
- PStore
- Registry Values
- Security
- SMTP
- System
- Terminal Server Install Mode
- User Interface
- Visual Studio
- Windows Secure Mode Policy
- Word Breaker

Develop with server technologies

2/18/2021 • 2 minutes to read • Edit Online

- Directory, Identity, and Access Services
- Rights Management
- System Administration
- Virtualization
- WMI/MI/OMI Providers
- Windows Server

Directory, Identity, and Access Services

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Directory Services
- Identity Services
- Access Management Services

Directory Services

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Directories
- Directory Access Technologies

Directories

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Active Directory Domain Services
- Active Directory Lightweight Directory Services
- Active Directory Schema
- Active Directory WMI Provider
- Microsoft UDDI SDK
- UDDI Class Library

Directory Access Technologies

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Active Directory Service Interfaces
- Directory Services Data Exchange
- DSML Services for Windows
- Lightweight Directory Access Protocol

Identity Services

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Microsoft Identity Manager 2016 Developer Reference
- Forefront Identity Manager 2010 R2 Developer Reference
- Forefront Identity Manager 2010 Developer Reference
- Identity Lifecycle Manager 2007
- Identity Lifecycle Manager 2007, Feature Pack 1 Developer Reference

Identity Lifecycle Manager 2007

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Certificate Lifecycle Manager Overview
- External Resources for CLM
- Certificate Lifecycle Manager Managed Reference

Access Management Services

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Active Directory Federation Services

Active Directory Federation Services

2/18/2021 • 2 minutes to read • Edit Online

Active Directory Federation Service (AD FS) is an authentication and authorization service that runs on Windows Server. AD FS provides users with single sign-on access between enterprise environments and web applications.

In this section

Active Directory Federation Services Overview

Overview information about AD FS.

ADFS Class Library

APIs used to develop application that use AD FS.

Related topics

Access Management Services

Microsoft Rights Management SDKs

2/18/2021 • 2 minutes to read • Edit Online

Three generations of Rights Management SDK are now available: Rights Management SDK 4.2, Microsoft Rights Management SDK 2.1 and Active Directory Rights Management Services SDK, as well as a scripting API, Active Directory Rights Management Services Scripting API, that enables custom administration of an RMS server.

In this section

Code Samples and Tools

A collection of Microsoft supplied RMS code samples and developer support tools across all supported operating systems; Android, iOS/OS X, Windows Phone and Windows Desktop.

RMS SDK 4.2 is a simplified, next-generation tool set that provides a lightweight development experience in enabling your Android, iOS, and Mac OS X device apps with information protection via Microsoft Rights Management services.

RMS SDK 2.1 is a powerful SDK offering for Windows desktop application developers and server based solution providers to enable their products with rights management.

Active Directory Rights Management Services SDK

NOTE

AD RMS SDK leveraging functionality exposed by the client in Msdrm.dll is available for use in Windows Server 2012, Windows 8, Windows Server 2008 R2, Windows 7, Windows Server 2008, and Windows Vista. It may be altered or unavailable in subsequent versions. Instead, use Microsoft Rights Management Services SDK 2.1, which leverages functionality exposed by the client in Msipc.dll.

Active Directory Rights Management Services Scripting API

The Windows Active Directory Rights Management Services (AD RMS) Scripting API can be used to create scripts to administer an AD RMS installation.

RMS Scenarios, Code and Tools

2/18/2021 • 2 minutes to read • Edit Online

This roadmap provides Microsoft supplied code samples that show you how it's done and code snippets that you can use in your applications.

ITEM	OPERATING SYSTEM	SUPPORTING SDK VERSION	DESCRIPTION
Read PFILE protected PDF	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**Read PFILE protected PDF** is a simple code example on our RMS Developer's Corner blog that uses the MSIPC File API to decrypt and open a PFILE protected PDF document.
IpcManagedAPI	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcManagedAPI** is a .NET (C#) representation of RMS SDK 2.1 to make it easy for your managed application to be RMS-enabled.
IPCNotepad	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IPCNotepad** is a sample RMS-enabled application that takes you through the basic steps that each RMS-enabled application should perform when protecting and consuming restricted content.
IpcDlp	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcDlp** is a sample RMS-enabled Data Leak Protection (DLP) application that takes you through the basic steps that a DLP RMS-enabled application should perform by using File API for protecting and consuming restricted content.
IpcAzureApp	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**IpcAzureApp** is a sample that demonstrates how to use RMS SDK in Azure application to protect data in Azure Blob Storage.
RmsDocumentInspector	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**RmsDocumentInspector** is a tool can give information about any RMS protected file such as content-id or user rights.

ITEM	OPERATING SYSTEM	SUPPORTING SDK VERSION	DESCRIPTION
RmsFileWatcher	Windows Desktop	RMS SDK 2.1 and later versions of the 2.x SDK	**RmsFileWatcher** is a sample that demonstrates how to build a Windows application that watches directories in the file system and applies RMS protection policies on every change, for example file added or file modified.
iOS/OS X usage scenarios	iOS / OS X	RMS SDK 4.2 and later versions of the 4.x SDK	**Objective C** code examples representing important development scenarios to get you accustomed to the RMS SDK. Examples include use of Microsoft Protected File format, custom protected file formats, and custom UI controls.
UI Library and Sample app	iOS	RMS SDK 4.2 and later versions of the 4.x SDK	**UI libraries and sample app for iOS** at GitHub, so you can get started quickly and re-use our standard UI in your apps.
UI Library and Sample app	Android	RMS SDK 4.2 and later versions of the 4.x SDK	**UI libraries and sample app for Android** at GitHub, so you can get started quickly and re-use our standard UI in your apps.
Android usage scenarios	Android	RMS SDK 4.2 and later versions of the 4.x SDK	**Java** code examples representing important development scenarios to get you accustomed to the RMS SDK. Examples include use of Microsoft Protected File format, custom protected file formats, and custom UI controls.

System Administration

2/18/2021 • 3 minutes to read • Edit Online

NOTE

These topics are for software developers. For customer support issues, see Support - Windows Help.

In this section

Boot Configuration Data WMI Provider

The Boot Configuration Data (BCD) Windows Management Instrumentation (WMI) provider provides programmatic access to BCD stores, which describe boot applications and boot application settings.

Group Policy

Group Policy enables policy-based administration using Microsoft Active Directory directory services. Group Policy uses directory services and security group membership to provide flexibility and support extensive configuration information.

Microsoft Management Console (MMC)

Microsoft Management Console (MMC) provides a simple, consistent, and integrated administration user interface and administration model.

Mobile Device Management Application Provider

The Mobile Device Management (MDM) application provider manages applications on devices that are subscribed to a MDM service.

Mobile Device Management Registration

MDM Registration enrolls devices into a MDM service.

Mobile Device Management Settings Provider

The MDM Settings Provider enables management of settings on devices subscribed to a MDM service.

NetShell

NetShell is a command line based tool that enables administrators to remotely administer and configure critical network services.

Settings Management Infrastructure

Settings Management Infrastructure (SMI) provides a standardized infrastructure to access and manipulate settings that are modifiable by users and applications.

Software Inventory Logging

Software Inventory Logging collects licensing data about software installed on a Windows Server, and provides remote access to the data so it can be aggregated easily by a datacenter.

Software Licensing API

The Software Licensing API (SLAPI) can be used to determine a genuine Microsoft Windows installation, install and log an asset management license, and retrieve information about the licensing policy of a software component.

System Restore

System Restore automatically monitors and records key system changes on a user's computer. It is designed to reduce support costs and increase customer satisfaction by enabling a user to undo a change that may have caused a problem with the system, or revert to a day when the system was performing optimally.

System Shutdown

The system shutdown functions and messages allow applications to log off the current user, shut down the system, or lock the workstation.

Task Scheduler

The Task Scheduler enables you to automatically perform routine tasks on a chosen computer. The Task Scheduler does this by monitoring whatever criteria you choose to initiate the tasks (referred to as triggers) and then executing the tasks when the criteria is met.

User Access Logging

User Access Logging (UAL) is a common framework for Windows Server roles to report their respective consumption metrics. This UAL framework is a foundational and critical component of the larger licensing management solution.

Windows Deployment Services

Windows Deployment Services (WDS) is the revised version of Remote Installation Services (RIS). WDS enables the deployment of Windows operating systems. You can use WDS to set up new clients with a network-based installation without requiring that administrators visit each computer or install directly from CD or DVD media.

Windows Genuine Advantage API

The Windows Genuine Advantage API is used to determine whether the Windows operating system that is running on the current system is a genuine copy.

Windows Management Instrumentation

Windows Management Instrumentation (WMI) is the infrastructure for management data and operations on Windows-based operating systems. You can write WMI scripts or applications to automate administrative tasks on remote computers.

Windows Defender WMIv2 APIs

Windows Defender WMI APIs can be used to manage malware protection through scripts or applications.

Windows PowerShell

Windows PowerShell is a task-based command-line shell and scripting language designed especially for system administration. Built on the .NET Framework, Windows PowerShell helps IT professionals and power users control and automate the administration of the Windows operating system and applications that run on Windows.

Windows Remote Management

The Windows Remote Management (WinRM) is the Microsoft implementation of WS-Management protocol, a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate. The WS-Management protocol specification provides a common way for systems to access and exchange management information across an IT infrastructure.

Windows Resource Protection

Windows Resource Protection (WRP) prevents the replacement of essential system files, folders, and registry keys that are installed as part of the operating system.

Windows Server Update Services

System administrators can use the Windows Server Update Services (WSUS) API to determine which updates apply to a computer or group of computers, download those updates, and install them with little or no user intervention.

Independent software vendors and end-user developers can integrate WSUS features into computer management or update management software to provide a seamless operating environment.

Windows System Assessment Tool

How to use WinSAT to assess the performance characteristics and capabilities of a computer.

Work Folders

Work Folders allows users to store and access work files on personal computers and devices.

Group Policy

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Group Policy
- Group Policy Management Console
- GPMC Class Library

Microsoft Management Console (MMC)

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Microsoft Management Console 3.0
- Microsoft Management Console 2.0

Microsoft Management Console 3.0

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Microsoft Management Console 3.0
- MMC Class Library

Windows Server Update Services

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Windows Server Update Services 3.0 Class Library
- Windows Server Update Services 2.0 Class Library
- Windows Update Agent API

Windows Server Update Services 2.0 Class Library

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Microsoft.UpdateServices.Administration Namespace (2.0)
- WSUS Samples

Virtualization

2/18/2021 • 2 minutes to read • Edit Online

In this section

Microsoft Virtual Server

Microsoft Virtual Server 2005 allows you to create separate virtual machines on top of your Microsoft Windows desktop, where you can install virtually any Intel-architecture operating system. Each virtual machine emulates a complete hardware system—from processor to network card—in a self-contained, isolated software environment, enabling the simultaneous operation of otherwise incompatible systems.

Windows Virtual PC

Windows Virtual PC is the latest Microsoft virtualization technology; it enables you to run many productivity applications on a virtual Windows environment, with a single click, directly from Windows 7. You can create separate virtual machines on top of your Windows 7 desktop. Each virtual machine emulates a complete hardware system—from processor to network card—in a self-contained, isolated software environment, enabling the simultaneous operation of otherwise incompatible systems.

Related topics

Hyper-V WMI Provider

WMI/MI/OMI Providers

2/18/2021 • 10 minutes to read • Edit Online

Windows Management Infrastructure (WMI), Management Instrumentation (MI) and Open Management Infrastructure (OMI) all use Management Object Format (MOF) files to describe the information made available through their respective providers.

Active Directory

The Active Directory provider, also known as the Directory Services (DS) provider, maps Active Directory objects to WMI. By accessing the Lightweight Directory Access Protocol (LDAP) namespace in WMI, you can reference or make an object an alias in the Active Directory.

Application Inventory

The WMI classes for Application Inventory enable discovery of the installed Win32 applications and Windows store applications on a Windows system.

Application Proxy

Application Proxy WMI Provider enables developers to access the Web Application Proxy service so administrators can publish applications for external access. Web Application Proxy is a reverse proxy for Active Directory Federation Services (AD FS).

BitLocker Drive Encryption (BDE)

Provides configuration and management for an area of storage on a hard disk drive, represented by an instance of **Win32_EncryptableVolume**, that can be protected by using encryption.

BITS

The Background Intelligent Transfer Service (BITS) Compact Server with BITS Remote Management allows authenticated administrators or controller applications to create, modify and manage BITS transfer jobs remotely without using the Internet Information Services (IIS) service.

BizTalk

Supplies access to the BizTalk administration objects represented by WMI classes.

Boot Configuration Data

The Boot Configuration Data (BCD) provider provides a store that is used to describe boot applications and boot application settings.

Boot Event Collector

The Boot Event Collector WMI provider provides access to connection and configuration information for the Setup and Boot Event Collection feature on Windows Server.

CIMWin32, Win32, Power Management Events

The CIMWin32 providers support the classes implemented in CimWin32.dll, and consist of the core CIM classes, the Win32 implementation of those classes, and power management events.

CIMWin32a

The CIMWin32a provider extends the classes available in the CIMWin32.

DcbQosCim

The DcbQosCim provider supports classes that describe Network QOS setting data, control setting data, and traffic class setting data.

Distributed File System (DFS)

The DFS provider supplies scriptable DFS functions through WMI.

Distributed File System Replication (DFSR)

Creates tools for configuring and monitoring the Distributed File System (DFS) service. For more information, see DFSR WMI Classes.

Dfsncimprov

The Dfsncimprov provider supports classes that implement DFS namespace access.

DhcpServerPSProvider

The DhcpServerPSProvider provider supports classes that interact with a dynamic host configuration protocol (DHCP) server.

Disk Quota

The Windows Disk Quota provider allows administrators to control the amount of data that each user stores on an NTFS file system.

Distributed Transaction Coordinator (DTC)

The DTC provider enables management of the DTC.

DNS

Enables administrators and programmers to configure Domain Name System (DNS) resource records (RRs) and DNS Servers using WMI.

Dnsclientcim Provider Classes

The Dnsclientcim provider supports classes that interact with a Domain Name System (DNS) client.

DnsClientPSProvider

The DnsClientPSProvider provider supports WMI classes that interact with a DNS client.

DnsServerPSProvider

The DnsServerPSProvider provider supports WMI classes that interact with a DNS server.

Event Log

The Event Log provider supplies access to data from the event log service, and notification of events.

Event Tracing Management

The Event Tracing Management provider provides access to Event Tracing for Windows (ETW) autologger session configurations and trace events.

Failover Cluster-Aware Updating

The Failover Cluster-Aware Updating (CAU) provider supports coordination with and management of CAU.

Failover Clustering Hyper-V

The Failover Clustering Hyper-V provider provides management and reporting of Hyper-V in a clustering environment.

Failover Clustering Storage QoS

The Failover Clustering Storage Quality of Service (QoS) provider provides management and reporting of the clustering storage QoS policies.

Failover Cluster V1 Provider

The Failover Cluster V1 provider provides management of a failover cluster.

Failover Cluster V1 Extensions

The Failover Cluster Extensions provider provides additional management of a failover cluster.

Gateway Health Monitor

The Gateway Health Monitor provider manages gateway health monitoring events and information.

Group Policy API

The Group Policy provider enables policy-based administration using Microsoft Active Directory (AD) directory services.

Host Guardian Service

The Host Guardian Service provider provides management of the Host Guardian Service for shielded VMs.

Hyper-V

The Hyper-V provider allows you to manage and retrieve information about virtual machines.

Hyper-V (V2)

The Hyper-V (V2) provider extends the Hyper-V provider.

Internet Information Services (IIS)

Exposes programming interfaces that can be used to query and configure the IIS metabase.

Internet Protocol Address Management (IPAM) Server

The IPAM Server Provider enables developers to manage IPAM through WMI.

IP Route Provider

The preinstalled IP Route provider supplies IPV4 network routing information, including (but not limited to) the information available through the **route print** command.

Intelligent Platform Management Interface (IPMI)

Supplies IPMI data from Baseboard Management Controller (BMC) operations.

iSCSI Target Server

The iSCSI Target Server provider supports a WMI interface for managing the Microsoft iSCSI Target Server, such as creating virtual disks, and for presenting them to the client.

Job Object

The Job Object provider supports access to data about named kernel job objects.

Kernel Trace

The preinstalled Kernel Trace event provider allows you to see kernel trace events on Process creation, Process termination, Thread creation, Thread termination and module load.

Live Communications Server 2003

Provides classes that create, register, configure, manage custom Session Initiation Protocol (SIP) applications with the Live Communications Server 2003.

Management Tools Registry

The Management Tools Registry provider provides remote access to the registry.

Management Tools Task Manager

The Management Tools Task Manager provider provides access and management of the Task Manager data.

MDM Application

The Mobile Device Management (MDM) application provider manages applications on devices that are subscribed to the MDM service.

MDM Bridge

The MDM Bridge provider enables MDM management of a network bridge.

MDM Settings

The MDM Settings Provider enables management of settings on devices enrolled with a MDM service.

MSFT_PCSVDevice

The MSFT_PCSVDevice provider exposes a class that defines a view class for a physical computer system.

MsNetImPlatform

The MsNetImPlatform provider section provides reference information for MsNetImPlatform provider classes implemented in NdisIMPlatCim.dll.

NetAdapterCim

The NetAdapterCim provider supports classes that access network adapters.

NetDaCim

This section provides reference information for NetDaCim Provider classes.

NetNcCim

Provides reference information for NetNcCim provider classes.

NetPeerDist

The NetPeerDist provider supports classes that interact with the Branch Cache infrastructure.

NetQosCim

The NetQosCim provider supplies data for network quality of service (QoS) and QoS setting data.

NetSwitchTeam

This section provides reference information for NetSwitchTeam provider classes declared in NetSwitchTeam.mof and implemented in NetSwitchTeamCim.dll.

NetTCPIP

The NetTCPIP provider supports classes that interact with TCPIP connections.

NetTtCim

This section provides reference information for NetTtCim provider classes defined in NetTtCim.mof and implemented in NetTtCim.dll.

NetWNV

The NetWNV provider supports classes that interact with net virtualization technologies.

Network Access Protection

The Network Access Protection provider exposes a platform for protected access to private networks.

Network Load Balancing

The Network Load Balancing (NLB) Provider enables management of a NLB cluster.

NetworkController Server

The provider enables management of a network controller server.

NFS

The provider for NFS allows you to create tools and scripts for configuring and monitoring the Windows Network File System.

Ping

The Ping provider supplies access to the status information provided by the standard ping command.

Policy

Provides extensions to group policy, and permits refinements in the application of policy.

Power Meter

The Power Meter provider supports the Power Metering and Budgeting (PMB) interface. These classes are the primary interface for the query of Power Meter Interface (PMI) information from underlying power meters on the system.

Power Policy

The Power Policy provider provides classes the ability to remotely manage all the power policy infrastructure.

RAMgmtPSProvider

The RAMgmtPSProvider provider provides classes to manage Remote Access.

RAServerPSProvider

The RAServerPSProvider provider provides classes to manage the Remote Access Server.

ReliabilityMetricsProvider

The ReliabilityMetricsProvider provider exposes system and Windows Event Log reliability metrics.

Remote Desktop Services

Enables consistent server administration in a Remote Desktop Services environment.

Reporting Services

Defines classes that allow you to write scripts and code to modify the settings of the report server and the Report Manager.

Resultant Set of Policy (RSoP)

Supplies methods to plan and debug policy settings in a what-if situation. These methods allow administrators to determine easily the combination of policy settings that apply to, or will apply to, a user or computer. This is known as the Resultant Set of Policy (RSoP). For more information, see About the RSoP WMI Method Provider and RSoP WMI Classes.

Security

Retrieves or changes security settings that control ownership, auditing, and access rights to files, directories, and shares.

ServerManager.DeploymentProvider

The ServerManager.DeploymentProvider exposes deployment functionality.

Session

Manages network sessions and connections.

Shadow Copy

Supplies management functions for the Shadow Copies of the Shared Folders feature.

Shielded VM Provisioning

The Shielded VM Provisioning provider enables a fabric controller to start the secure provisioning of a shielded VM on a Hyper-V host.

Shielded VM Provisioning Service

The provider enables provisioning of a shielded virtual machine.

SMB Management

The SMB Management API provides classes and methods to manage shares and share access.

SNMP

Maps Simple Network Management Protocol (SNMP) objects that are defined in Management Information Base (MIB) schema objects to classes. For more information, see Setting up the WMI SNMP Environment.

Software Inventory Logging

The Software Inventory Logging provider collects licensing data about software installed on a Windows Server, and provides remote access to the data so it can be aggregated easily by a datacenter.

Software Licensing for Windows Vista

Software Licensing Classes used for Windows Vista.

Software License

The Software Licensing provider retrieves software licensing data.

Storage Volume

The Storage Volume provider supplies volume management functions.

Storage Replica

The provider enables management of a storage replica.

System Registry

The System Registry provider enables management applications to retrieve and modify data in the system registry, and receive notifications when changes occur.

System Restore

Supplies classes that configure and use System Restore functionality. For more information, see Configuring System Restore and the System Restore WMI Classes.

Trusted Platform Module

Provides access to data about a security device, represented by an instance of **Win32_TPM**, that is the root of trust for a Microsoft Windows trusted platform computer system.

Trustmon

The Trustmon provider is an instance provider that creates classes with information about the trust relationships between domains.

User Access Logging

User Access Logging (UAL) is a common framework for Windows Server roles to report their respective consumption metrics.

UserProfileProvider

The UserProfileProvider provider exposes classes that provide information about a user profile on a Windows system, as well as the health status of a redirected user folder.

User State Management

The User State Management provider exposes a management and reporting API for enterprise scenarios.

View

Creates new instances and methods based on instances of other classes. Two versions of the View provider are available on 64-bit platforms.

VPNClientPSProvider

The VPNClientPSProvider provider exposes a platform for automating connectivity to a virtual private network client.

WDM

Provides access to the classes, instances, methods, and events of hardware drivers that conform to the Windows Driver Model (WDM).

WFasCim

The WFasCim provider exposes network security and filtering features.

WhqlProvider

The WhqlProvider provider exposes digital signature information about drivers.

Win32ClockProvider

The Win32ClockProvider provider exposes the current, local, and UTC-based timestamps on a computer system.

Windows Data Access Components (WDAC)

Provides management of WDAC.

Windows Defender

The Windows Defender provider exposes Windows Defender security features.

Windows Installer

The Windows Installer provider, also known as the MSI provider allows applications to access information collected from Windows Installer compliant applications.

Windows Product Activation

The Windows Product Activation (WPA) provider is an anti-piracy technology aimed at reducing the casual copying of software.

Windows Server Manager

The provider enables access to and management of the configuration controlled by the Server Manager application.

Windows Server Storage Management (MsftStrgMan)

The MsftStrgMan provider provides management for storage systems on Windows Server products.

Windows Storage Management (StrgMgmt)

The StrgMgmt provider can be used to manage a wide range of storage configurations, from tablets to external storage arrays on servers.

Windows System Assessment Tool

The Windows System Assessment Tool (WinSAT) exposes a number of classes that assesses the performance characteristics and capabilities of a computer.

WMI Core

The WMI Core provider defines classes that compose the core functionality of WMI.

Msft_ProviderSubSystem

The Msft_ProviderSubSystem provider supports providers.

Win32_Perf

The **Win32_Perf** abstract class is the base class for the performance counter classes **Win32_PerfRawData** and **Win32_PerfFormattedData**. It defines the required timestamp and frequency properties used in the CounterType algorithms for the performance counter classes.

Win32_PerfFormattedData

An abstract base class for the pre-installed, calculated data classes.

Win32_PerfRawData

The performance counter class **Win32_PerfRawData** is the abstract base class for all concrete raw performance counter classes. To appear in System Monitor, performance counter classes must be added to the "Root\CIMv2" namespace and derived from **Win32_PerfRawData**.

WMIPerfClass

Creates the WMI Performance Counter Classes. Data is dynamically supplied to these performance classes by the WMIPerfInst provider.

WmiPerfInst

Supplies raw and formatted performance counter data dynamically from Performance Counter Class definitions.

Work Folders

Work Folders is used to synchronize files with multiple PCs and mobile devices.

Windows Server

2/18/2021 • 2 minutes to read • Edit Online

Windows Server is a group of operating systems designed by Microsoft that supports enterprise-level management, data storage, applications, and communications. Previous versions of Windows Server have focused on stability, security, networking, and various improvements to the file system. Other improvements also have included improvements to deployment technologies, as well as increased hardware support. Microsoft has also created specialized SKUs of Windows Server that focus on the home and small business markets. Windows Server 2012 R2 is the latest release of Windows Server, and focuses on cloud computing.

Windows Server Versions

The following table describes the current documented versions of Windows Server.

SERVER VERSION	DESCRIPTION
Windows Server 2012 R2 and Windows Server 2012	Emphasizes cloud support with features such as improved IP-addressing, updated Hyper-V, and a new file system (ReFS). Windows Server 2012 R2 includes enhancements to virtualization, management, storage, networking, virtual desktop infrastructure, access protection, information protection, web services, and application platform infrastructure.
Windows Server 2008 R2 and Windows Server 2008	Includes a number of additional security and administrative features shared with Windows Vista: a re-written networking stack, improved Firewall, additional .NET Framework technology, and numerous improvements to the kernel, file, and memory systems.
Windows Server 2003 R2 and Windows Server 2003	Integrates a number of features from Windows XP. Includes improved networking installation and integration, improved web services, and increased capabilities for DTFS.

Other Windows Server SKUs

The following table describes the specialized versions of Windows Server.

SKU	DESCRIPTION
Windows Server Essentials 2012 and 2012 R2	Continuation of the Small Business Server line, and focuses on a tight integration with the unified Management Console. Based off of Windows Server 2012 and Windows Server 2012 R2.
Windows Essential Business Server 2008	Designed for medium-sized businesses, for a maximum of 300 users, and focused on integration with the Management Console. Based off of Windows Server 2008.

SKU	DESCRIPTION
Windows Small Business Server	Designed to be an integrated server suite for small-to-medium-sized businesses with no more than 75 users. Based off of Windows Server operating systems up to Windows Server 2008 R2.
Windows Home Server and Windows Home Server 2011	Designed for a home with multiple networked PCs, and emphasized file sharing, remote access, and automated backups. Based off of Windows Server 2003 and Windows Server 2008 R2.

Related topics

Windows Server Blog

Windows Server on TechNet

What's New: Windows Server 2012 R2 & Windows Server 2012

2/18/2021 • 4 minutes to read • Edit Online

The following list describes new and updated feature areas for Windows Server 2012 and Windows Server 2012 R2. For more information on new Windows 8 and Windows 8.1 technologies, see Windows 8 and Windows 8.1 Technologies.

What's New for Windows Server 2012 R2

- Data Deduplication

  Data deduplication finds and removes duplication within data on a volume while ensuring that the data remains correct and complete. This makes it possible to store more file data in less space on the volume. For Windows Server 2012 R2, a number of parameters and error codes were activated, and the **MSFT_DedupFileMetadata** class was added.

- Software Inventory Logging

  New! Software Inventory Logging collects licensing data about software installed on a Windows Server, and provides remote access to the data so it can be aggregated easily by a datacenter.

- iSCSI Target Server

  The iSCSI Target Server API provides Windows Management Instrumentation (WMI) providers for managing the Microsoft iSCSI Target Server, such as creating virtual disks, and for presenting them to the client. A number of new features were added for Windows Server 2012 R2.

- Mobile Device Management Registration

  New! An industry trend has been developing in which employees connect their personal mobile computing devices to the corporate network and resources (either on premise or through the cloud) to perform workplace tasks. This trend requires support for easy configuration of the network and resources, such that employees can register personal devices with the company for work-related purposes. Applications and technology to support easy configuration must also enable IT professionals to manage the risk associated with having uncontrolled devices connected to the corporate network. The Mobile Device Management (MDM) registration enrolls a device into a management service.

- Windows PowerShell

  Windows PowerShell is a task-based command-line shell and scripting language designed especially for system administration. New in Windows Server 2012 R2, Windows PowerShell v4 supports Desired State Configuration, which is a new management platform in Windows PowerShell that enables deploying and managing configuration data for software services and managing the environment in which these services run.

What's New for Windows Server 2012

- Windows Clustering

  Windows Clustering allows you to manage both network load-balanced clusters as well as failover clusters. A number of new features were added in this release, including new Group management features, common properties, and integration with WMI.

- User Access Logging

  New! User Access Logging (UAL) is a common framework for Windows Server roles to report their respective consumption metrics. This UAL framework is a foundational and critical component of the larger licensing management solution.

- Windows Remote Management

  Windows Remote Management (WinRM) is the Microsoft implementation of WS-Management Protocol, a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate. Windows Server 2012 includes a number of Connection API's that focus on interacting with running instances and shells.

- Windows Management Infrastructure

  New! The Windows Management Infrastructure (MI) features represent the latest version of the Windows Management Instrumentation (WMI) technologies, which are based on the CIM standard from DMTF (Distributed Management Task Force). MI is fully compatible with previous versions of WMI and provides a host of features and benefits that make designing and developing providers and clients easier than ever.

- Data Deduplication

  New! Data deduplication finds and removes duplication within data on a volume while ensuring that the data remains correct and complete. This makes it possible to store more file data in less space on the volume.

- iSCSI Target Server

  New! The iSCSI Target Server API provides Windows Management Instrumentation (WMI) providers for managing the Microsoft iSCSI Target Server, such as creating virtual disks, and for presenting them to the client.

- NFS Provider for WMI

  New! The NFS WMI provider provides a means of managing NFS server and client settings, file shares, netgroups, and client groups.

- Offline Files

  The Offline Files API allows applications to control and monitor the behavior of Offline Files programmatically. Windows Server 2012 adds the **OfflineFilesQueryStatusEx**, **OfflineFilesStart** and **RenameItem** functions.

- SMB Management

  New! The SMB Management API provides WMI classes and methods to manage shares and share access.

- Server Core

  Windows Server provides minimal server installation options for computers running on the Windows Server 2008 operating system or later. Windows Server 2012 offers Server Core as both a configuration as well as an installation options.

- Windows Server Backup

  The Windows Server Backup feature automatically backs up and restores application data. Windows Server 2012 includes the Cloud Backup Provider API.

- Windows Desktop Sharing

  Windows Desktop Sharing is a multiple-party screen-sharing technology. Windows Server 2012 implements this technology using the Windows Duplication API.

- Remote Desktop Services

  Windows Desktop Sharing allows the user to remotely access an instance of the operating system. Windows Server 2012 includes a number of new features, including Child sessions, RemoteFX Media redirection, and the Remote Desktop Broker client.

- Windows PowerShell

  Windows PowerShell is a task-based command-line shell and scripting language designed especially for system administration. New in Windows Server 2012, Windows PowerShell v3 supports Windows PowerShell Workflow, which leverages the benefits of Windows Workflow Foundation to allow the automation of long-running tasks.

- Management OData

  New! Also known as Windows PowerShell Web Services, Management OData, new in Windows PowerShell v3, allows the creation of an ASP.NET web service end point that exposes management data, acessed through Windows PowerShell commands, as OData web service entities.

Related topics

Windows Server 2012 R2 on TechNet

Windows Server 2012 R2 and Windows Server 2012 on TechNet Library

Windows Server 2012 R2 on Microsoft.Com

Mobile Device Management Registration

2/22/2020 • 2 minutes to read • Edit Online

Purpose

An industry trend has been developing in which employees connect their personal mobile computing devices to the corporate network and resources (either on premise or through the cloud) to perform workplace tasks. This trend requires support for easy configuration of the network and resources, such that employees can register personal devices with the company for work-related purposes. Applications and technology to support easy configuration must also enable IT professionals to manage the risk associated with having uncontrolled devices connected to the corporate network. The Mobile Device Management (MDM) registration enrolls a device into a management service.

In this section

MDM Registration Reference

The following programming elements are used with MDM Registration.

Developer audience

Mobile Device Management Registration is designed for use by companies developing and deploying MDM services.

System Restore

11/2/2020 • 2 minutes to read • Edit Online

Purpose

System Restore automatically monitors and records key system changes on a user's computer. It is designed to reduce support costs and increase customer satisfaction by enabling a user to undo a change that may have caused a problem with the system, or revert to a day when the system was performing optimally.

NOTE

The following documentation is targeted for developers. If you are an end-user looking for information on how to use System Restore, see What Is System Restore?

Developer audience

The System Restore API is designed for use by C/C++ programmers. A familiarity with Windows Management Instrumentation (WMI) is required to use the scripting interface.

Run-time requirements

The System Restore API is supported on client operating systems starting with Windows XP. For information about which operating systems are required to use a particular API element, see the Requirements section of its documentation.

In this section

TOPIC	DESCRIPTION
Overview	An overview of how System Restore works.
Reference	Documentation of System Restore functions, structures, and classes.
Samples	A sample program written in C.

Related topics

WMI

System Shutdown

2/18/2021 • 2 minutes to read • Edit Online

The system shutdown functions and messages allow applications to log off the current user, shut down the system, or lock the workstation:

- Shutdown Changes for Windows Vista
- About System Shutdown
- Using System Shutdown
- System Shutdown Reference

Task Scheduler for developers

2/22/2020 • 2 minutes to read • Edit Online

IMPORTANT

This topic and the other topics in this section are for a developer audience. If you're wishing to use the the Task Scheduler component in your capacity as an administrator, or an IT Professional, then see Task Scheduler.

About the Task Scheduler

The Task Scheduler enables you to automatically perform routine tasks on a chosen computer. Task Scheduler does this by monitoring whatever criteria you choose (referred to as triggers) and then executing the tasks when those criteria are met.

You can use the Task Scheduler to execute tasks such as starting an application, sending an email message, or showing a message box. Tasks can be scheduled to execute in response to these events, or triggers.

- When a specific system event occurs.
- At a specific time.
- At a specific time on a daily schedule.
- At a specific time on a weekly schedule.
- At a specific time on a monthly schedule.
- At a specific time on a monthly day-of-week schedule.
- When the computer enters an idle state.
- When the task is registered.
- When the system is booted.
- When a user logs on.
- When a Terminal Server session changes state.

Developers

The Task Scheduler provides APIs in these forms.

- Task Scheduler 2.0: interfaces and objects are provided for C++, and for scripting development, respectively.
- Task Scheduler 1.0: interfaces are provided for C++ development.

Run-time requirements

The Task Scheduler requires the following operating systems.

- Task Scheduler 2.0: Client requires Windows Vista, or above. Server requires Windows Server 2008, or above.
- Task Scheduler 1.0: Client requires Windows Vista, or Windows XP. Server requires Windows Server 2008, or Windows Server 2003.

In this section

TOPIC	DESCRIPTION
What's new in Task Scheduler	Summary of new functionality introduced by the Task Scheduler.
About the Task Scheduler	General conceptual information about the Task Scheduler API.
Using the Task Scheduler	Code examples that show how to use the Task Scheduler APIs.
Task Scheduler reference	Detailed reference information for Task Scheduler APIs and the Task Scheduler schema.

Windows Deployment Services

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Windows Deployment Services (WDS) is the revised version of Remote Installation Services (RIS). WDS enables the deployment of Windows operating systems. You can use WDS to set up new clients with a network-based installation without requiring that administrators visit each computer or install directly from CD or DVD media.

Developer audience

The primary developer audience of the WDS API is for groups that develop custom tools and processes for IT and other computer administration groups. In environments where the standard Windows Deployment Services (WDS) solution cannot be used, the WDS API enables programmatic access to some WDS components.

Original Equipment Manufacturers (OEMs), system builders, and corporate IT professionals looking for information on how to deploy Windows onto new computers, should see the information about the standard Windows Deployment Services (WDS) solution in the Windows Deployment Services Update Step-by-Step Guide and the Windows Automated Installation Kit (WAIK).

Run-time requirements

WDS is available as an add-on for Windows Server 2003 with Service Pack 1 (SP1) and is included in the operating system starting with Windows Server 2003 with Service Pack 2 (SP2) and Windows Server 2008. The WDS PXE Server API requires the WDS server role on the server to implement custom PXE providers. The WDS Client API requires the Microsoft Windows Preinstallation Environment (Windows PE 2.0) phase of setup processing. A RAMDISK bootable image of Windows PE 2.0 in the .WIM format must be downloaded as part of the network boot process to implement custom WDS clients.

In this section

TOPIC	DESCRIPTION
About the Windows Deployment Services API	General information about the WDS Server API and WDS Client API.
Windows Deployment Services Reference	Describes the Windows Deployment Services functions and structures.

Windows Management Instrumentation

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Windows Management Instrumentation (WMI) is the infrastructure for management data and operations on Windows-based operating systems. You can write WMI scripts or applications to automate administrative tasks on remote computers but WMI also supplies management data to other parts of the operating system and products, for example System Center Operations Manager, formerly Microsoft Operations Manager (MOM), or Windows Remote Management (WinRM).

NOTE

The following documentation is targeted for developers and IT administrators. If you are an end-user that has experienced an error message concerning WMI, you should go to Microsoft Support and search for the error code you see on the error message. For more information about troubleshooting problems with WMI scripts and the WMI service, see WMI Isn't Working!

NOTE

WMI is fully supported by Microsoft; however, the latest version of administrative scripting and control is available through the Windows Management Infrastructure (MI). MI is fully compatible with previous versions of WMI, and provides a host of features and benefits that make designing and developing providers and clients easier than ever. For more information, see Windows Management Infrastructure (MI).

Where applicable

WMI can be used in all Windows-based applications, and is most useful in enterprise applications and administrative scripts.

System administrators can find information about using WMI at the TechNet ScriptCenter, and in various books about WMI. For more information, see Further Information.

Developer audience

WMI is designed for programmers who use C/C++, the Microsoft Visual Basic application, or a scripting language that has an engine on Windows and handles Microsoft ActiveX objects. While some familiarity with COM programming is helpful, C++ developers who are writing applications can find good examples for getting started at Creating a WMI Application Using C++.

To develop managed code providers or applications in C# or Visual Basic .NET using the .NET Framework, see WMI in .NET Framework.

Many administrators and IT professionals access WMI through PowerShell. The Get-WMI cmdlet for PowerShell enables you to retrieve information for a local or remote WMI repository. As such, a number of topics and classes, especially in the Creating WMI Clients section, contain PowerShell examples. For additional information on using PowerShell, see Windows PowerShell and Scripting with Windows PowerShell.

Run-time requirements

For more information about which operating system is required to use a specific API element or WMI class, see the Requirements section of each topic in the WMI documentation.

If an expected component appears to be missing, see Operating System Availability of WMI Components.

You do not need to download or install a specific software development (SDK) in order to create scripts or applications for WMI. However, there are some WMI administrative tools that developers find useful. For more information, see the Downloads section in Further Information.

In this section

About WMI

General information about WMI.

Using WMI

Information about how to develop applications to use WMI, which includes information about tools.

WMI Reference

Documentation about the WMI classes, WMI C++ classes, WMI COM API, Scripting API, and other WMI reference material.

Boot Event Collector WMI Provider

2/18/2021 • 2 minutes to read • Edit Online

Purpose

The Boot Event Collector WMI provider provides access to connection and configuration information for the Setup and Boot Event Collection feature on Windows Server. This allows you to view a list of the current connections and the connection history between a collector server and its target computers. In addition, this provider allows you to manage the configuration of a collector server.

NOTE

The Boot Event Collector WMI provider is implemented in BEvtCol.exe.

In this section

TargetForwarding

Retrieves forwarding data from a target computer.

TargetForwardingDestination

The known destinations containing the collected data. Available only if the collector is running with the status log enabled.

TargetForwardingHistory

The recent history of changes to the forwarding data for a target computer.

Control

Control of the collector instance. Requires the Administrator (BA) privileges.

Hyper-V management

2/18/2021 • 2 minutes to read • Edit Online

Here we describe how to manage or get info about virtual machines.

In this section

TOPIC	DESCRIPTION
About Hyper-V management	This section contains info for Hyper-V management.
Hyper-V management reference	This section contains the reference for Hyper-V management.

MDM Bridge WMI Provider

11/2/2020 • 16 minutes to read • Edit Online

[Some information relates to pre-released product which may be substantially modified before it's commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here.]

Purpose

MDM Bridge WMI Provider enables mobile device management of a network bridge. This provider is defined in DMWmiBridgeProv.mof and implemented in DMWmiBridgeProv.dll.

In this section

TOPIC	DESCRIPTION
MDM_ActiveSync_User_Accounts01_01	The MDM_ActiveSync_User_Accounts01_01 class defines all available ActiveSync accounts.
MDM_ActiveSync_User_ContentTypes04_01	The MDM_ActiveSync_User_ContentTypes04_01 class defines the type of content to be individually enabled/disabled for sync.
MDM_ActiveSync_User_Options03	The MDM_ActiveSync_User_Options03 class defines the options that are set for each ActiveSync account.
MDM_AppLocker_ApplicationLaunchRestrictions01_EXE03	The MDM_AppLocker_ApplicationLaunchRestrictions01_EXE03 class allows you to specify which EXE applications are allowed to launch.
MDM_AppLocker_ApplicationLaunchRestrictions01_StoreApps03	The MDM_AppLocker_ApplicationLaunchRestrictions01_StoreApps03 class allows you to specify which EXE applications are allowed or disallowed for Enterprise Data Protection.
MDM_AppLocker_CodeIntegrity03	The MDM_AppLocker_CodeIntegrity03 class defines the policy for Code Integrity.
MDM_AppLocker_DLL03	The MDM_AppLocker_DLL03 class defines the policy restrictions for processing DLL files.
MDM_AppLocker_EnterpriseDataProtection01_EXE03	The MDM_AppLocker_EnterpriseDataProtection01_EXE03 class captures the list of executable applications that are allowed to handle enterprise data.
MDM_AppLocker_EnterpriseDataProtection01_StoreApps03	The MDM_AppLocker_EnterpriseDataProtection01_StoreApps03 class captures the list of Windows apps that are allowed to handle enterprise data. Should be used in conjunction with the settings in ./Vendor/MSFT/Policy/ConfigSource/DataProtection .

TOPIC	DESCRIPTION
MDM_AppLocker_MSI03	The MDM_AppLocker_MSI03 class defines the policy restrictions for processing MSI files.
MDM_AppLocker_Script03	The MDM_AppLocker_Script03 class defines the policy restrictions for processing DLL files.
MDM_AssignedAccess	The MDM_AssignedAccess class is used to set the device to run in kiosk mode.
MDM_ClientCertificateInstall_Install03	The MDM_ClientCertificateInstall_Install03 class enables the enterprise to set the installation of client certificates.
MDM_ClientCertificateInstall_PFXCertInstall01_01	The MDM_ClientCertificateInstall_PFXCertInstall01_01 class enables the enterprise to use unique IDs to differentiate different certificate install requests.
MDM_ClientCertificateInstall_SCEP01_01	The MDM_ClientCertificateInstall_SCEP01_01 class enables access to the node for SCEP certificate installation, using unique IDs to differentiate different certificate install requests.
MDM_DevDetail	The MDM_DevDetail class handles the management object which provides device-specific parameters to the OMA DM server.
MDM_DevDetail_Ext01	The MDM_DevDetail_Ext01 class handles the management object which provides device-specific parameters to the OMA DM server.
MDM_DevDetail_Microsoft02	The MDM_DevDetail_Microsoft02 class handles the management object which provides device-specific parameters to the OMA DM server.
MDM_DeviceStatus	The MDM_DeviceStatus class is used by the enterprise to keep track of device inventory and query the state of compliance of these devices with their enterprise policies.
MDM_DeviceStatus_Antispyware01	The MDM_DeviceStatus_Antispyware01 class is used by the enterprise to query the state of antispyware compliance of devices with their enterprise policies.
MDM_DeviceStatus_Antivirus01	The MDM_DeviceStatus_Antivirus01 class is used by the enterprise to query the state of antivirus compliance of devices with their enterprise policies.

TOPIC	DESCRIPTION
MDM_DeviceStatus_Battery01	The MDM_DeviceStatus_Battery01 class is used by the enterprise to query the state of battery compliance of devices with their enterprise policies.
MDM_DeviceStatus_CellularIdentities01_01	The MDM_DeviceStatus_CellularIdentities01_01 class allows you to query whether device are in compliance with enterprise encryption policy.
MDM_DeviceStatus_Compliance01	The MDM_DeviceStatus_Compliance01 class allows you to query whether device are in compliance with enterprise encryption policy.
MDM_DeviceStatus_Firewall01	The MDM_DeviceStatus_Firewall01 class is used by the enterprise to query the state of firewall compliance of devices with their enterprise policies.
MDM_DeviceStatus_NetworkIdentifiers01_01	The MDM_DeviceStatus_NetworkIdentifiers01_01 class allows you to query a device for network properties.
MDM_DeviceStatus_OS01	The MDM_DeviceStatus_OS01 class is used by the enterprise to query the operating system on devices with their enterprise policies.
MDM_DeviceStatus_TPM01	The MDM_DeviceStatus_TPM01 class is used by the enterprise to query the TPM version of devices with their enterprise policies.
MDM_DeviceStatus_UAC01	The MDM_DeviceStatus_UAC01 class is used by the enterprise to query the UAC status of devices with their enterprise policies.
MDM_EnterpriseAPN_01	The MDM_EnterpriseAPN_01 class is used by the enterprise to provision an APN for the Internet.
MDM_EnterpriseAPN_Settings01	The MDM_EnterpriseAPN_Settings01 class is used by the enterprise to change APN global settings.
MDM_EnterpriseDataProtection	The MDM_EnterpriseDataProtection class is used to determine the current status of Windows Information Protection (WIP) (formerly known as Enterprise Data Protection) specific settings.
MDM_EnterpriseDataProtection_Settings01	The MDM_EnterpriseDataProtection_Settings01 class is used to configure Windows Information Protection (WIP) (formerly known as Enterprise Data Protection) specific settings.
MDM_EnterpriseModernAppManagement_AppInstallation01_01	The MDM_EnterpriseModernAppManagement_AppInstallation01_01 class is used to install apps from the Windows Store or a hosted location.

TOPIC	DESCRIPTION
MDM_EnterpriseModernAppManagement_AppManagement01	The MDM_EnterpriseModernAppManagement_AppManagement01 class starts the Windows Update scan and reports the last scan error.
MDM_EnterpriseModernAppManagement_AppManagement01_02	The MDM_EnterpriseModernAppManagement_AppManagement01_02 class specifies whether you want to block a specific app from being updated via auto-updates.
MDM_EnterpriseModernAppManagement_AppManagement01_03	The MDM_EnterpriseModernAppManagement_AppManagement01_03 class provides specific information about the app, such as name, version, and publisher.
MDM_EnterpriseModernAppManagement_AppSettingPolicy04	The MDM_EnterpriseModernAppManagement_AppSettingPolicy04 class specifies all managed app setting values.
MDM_EnterpriseModernAppManagement_StoreLicenses02_01	The MDM_EnterpriseModernAppManagement_StoreLicenses02_01 class is used to manage licenses for store apps.
MDM_HealthAttestation	The MDM_HealthAttestation class enables enterprise IT managers to assess the health of managed devices and take enterprise policy actions.
MDM_PassportForWork	The MDM_PassportForWork is used to provision Windows Hello for Business.
MDM_PassportForWork_Biometrics01	The MDM_PassportForWork_Biometrics01 class defines biometric settings.
MDM_PassportForWork_Device_Policies02	The MDM_PassportForWork_Device_Policies02 class defines the Windows Hello for Business policy settings.
MDM_PassportForWork_PINComplexity03	The MDM_PassportForWork_PINComplexity03 class defines the PIN complexity for the logon credentials for Windows Hello for Business.
MDM_PassportForWork_Policies02	The MDM_PassportForWork_Policies02 class provisions Windows Hello for Business.
MDM_PassportForWork_Remote03	The MDM_PassportForWork_Remote03 class defines the Windows Hello for Business remote policy settings.
MDM_Policy_Config01_AboveLock02	The MDM_Policy_Config01_AboveLock02 class represents policies that determine actions that are allowed above the Device Lock screen.
MDM_Policy_Config01_Accounts02	The MDM_Policy_Config01_Accounts02 class represents policies related to accounts.

TOPIC	DESCRIPTION
MDM_Policy_Config01_ApplicationManagement02	The MDM_Policy_Config01_ApplicationManagement02 class represents the application management policies available.
MDM_Policy_Config01_Authentication02	The MDM_Policy_Config01_Authentication02 class represents the authentication management policies available.
MDM_Policy_Config01_BitLocker02	The MDM_Policy_Config01_Bitlocker02 class represents the BitLocker policies available.
MDM_Policy_Config01_Bluetooth02	The MDM_Policy_Config01_Bluetooth02 class represents the Bluetooth management policies available.
MDM_Policy_Config01_Browser02	The MDM_Policy_Config01_Browser02 class represents the browser policies available.
MDM_Policy_Config01_Camera02	The MDM_Policy_Config01_Camera02 class represents the camera policies available.
MDM_Policy_Config01_Connectivity02	The MDM_Policy_Config01_Connectivity02 class represents the connection policies available.
MDM_Policy_Config01_Cryptography02	The MDM_Policy_Config01_Cryptography02 class represents policies related to cryptography.
MDM_Policy_Config01_DataProtection02	The MDM_Policy_Config01_DataProtection02 class represents the data protection policies available.
MDM_Policy_Config01_Defender02	The MDM_Policy_Config01_Defender02class represents policies related to Windows Defender
MDM_Policy_Config01_DeliveryOptimization02	The MDM_Policy_Config01_DeliveryOptimization02 class represents the delivery optimization policies available.
MDM_Policy_Config01_DeviceLock02	The MDM_Policy_Config01_DeviceLock02 class represents the device lock policies available.
MDM_Policy_Config01_Experience02	The MDM_Policy_Config01_Experience02 class represents the experience policies available.
MDM_Policy_Config01_Licensing02	The MDM_Policy_Config01_Licensing02 class represents the licensing policies available.
MDM_Policy_Config01_LockDown02	The MDM_Policy_Config01_Lockdown02 class represents the lockdown policies available.
MDM_Policy_Config01_Maps02	The MDM_Policy_Config01_Maps02 class represents the map policies available.
MDM_Policy_Config01_NetworkIsolation02	The MDM_Policy_Config01_NetworkIsolation02 class represents the browser policies available.

TOPIC	DESCRIPTION
MDM_Policy_Config01_Privacy02	The MDM_Policy_Config01_Privacy02 class represents the search policies available.
MDM_Policy_Config01_Search02	The MDM_Policy_Config01_Search02 class represents the search policies available.
MDM_Policy_Config01_Security02	The MDM_Policy_Config01_Security02 class represents the security policies available.
MDM_Policy_Config01_Start02	The MDM_Policy_Config01_Start02 class represents the start screen policies available.
MDM_Policy_Config01_System02	The MDM_Policy_Config01_System02 class represents the System policies available.
MDM_Policy_Config01_TextInput02	The MDM_Policy_Config01_TextInput02 class represents the text input policies available.
MDM_Policy_Config01_Update02	The MDM_Policy_Config01_Update02 class represents the update policies available.
MDM_Policy_Config01_WiFi02	The MDM_Policy_Config01_WiFi02 class represents the Wi-Fi policies available.
MDM_Policy_Config01_WindowsInkWorkspace02	The MDM_Policy_Config01_WindowsInkWorkspace02 class represents the ink workspace policies available.
MDM_Policy_Config01_WirelessDisplay02	The MDM_Policy_Config01_WirelessDisplay02 class represents the wireless display policies available.
MDM_Policy_Result01_AboveLock02	The MDM_Policy_Result01_AboveLock02 class represents policies that determine actions that are allowed above the Device Lock screen.
MDM_Policy_Result01_Accounts02	The MDM_Policy_Result01_Accounts02 class represents policies related to accounts.
MDM_Policy_Result01_ApplicationManagement02	The MDM_Policy_Result01_ApplicationManagement02 class represents the application management policies available.
MDM_Policy_Result01_Authentication02	The MDM_Policy_Result01_Authentication02 class represents the authentication management policies available.
MDM_Policy_Result01_BitLocker02	The MDM_Policy_Result01_Bitlocker02 class represents the BitLocker policies available.
MDM_Policy_Result01_Bluetooth02	The MDM_Policy_Result01_Bluetooth02 class represents the Bluetooth management policies available.
MDM_Policy_Result01_Browser02	The MDM_Policy_Result01_Browser02 class represents the browser policies available.

TOPIC	DESCRIPTION
MDM_Policy_Result01_Camera02	The MDM_Policy_Result01_Camera02 class represents the camera policies available.
MDM_Policy_Result01_Connectivity02	The MDM_Policy_Result01_Connectivity02 class represents the connection policies available.
MDM_Policy_Result01_Cryptography02	The MDM_Policy_Result01_Cryptography02 class represents policies related to cryptography.
MDM_Policy_Result01_DataProtection02	The MDM_Policy_Result01_DataProtection02 class represents the data protection policies available.
MDM_Policy_Result01_Defender02	The MDM_Policy_Result01_Defender02class represents policies related to Windows Defender
MDM_Policy_Result01_DeliveryOptimization02	The MDM_Policy_Result01_DeliveryOptimization02 class represents the delivery optimization policies available.
MDM_Policy_Result01_DeviceLock02	The MDM_Policy_Result01_DeviceLock02 class represents the device lock policies available.
MDM_Policy_Result01_Experience02	The MDM_Policy_Result01_Experience02 class represents the experience policies available.
MDM_Policy_Result01_Licensing02	The MDM_Policy_Result01_Licensing02 class represents the licensing policies available.
MDM_Policy_Result01_LockDown02	The MDM_Policy_Result01_Lockdown02 class represents the lockdown policies available.
MDM_Policy_Result01_Maps02	The MDM_Policy_Result01_Maps02 class represents the map policies available.
MDM_Policy_Result01_NetworkIsolation02	The MDM_Policy_Result01_NetworkIsolation02 class represents the browser policies available.
MDM_Policy_Result01_Privacy02	The MDM_Policy_Result01_Privacy02 class represents the search policies available.
MDM_Policy_Result01_Search02	The MDM_Policy_Result01_Search02 class represents the search policies available.
MDM_Policy_Result01_Security02	The MDM_Policy_Result01_Security02 class represents the security policies available.
MDM_Policy_Result01_Start02	The MDM_Policy_Result01_Start02 class represents the start screen policies available.
MDM_Policy_Result01_System02	The MDM_Policy_Result01_System02 class represents the System policies available.
MDM_Policy_Result01_TextInput02	The MDM_Policy_Result01_TextInput02 class represents the text input policies available.

TOPIC	DESCRIPTION
MDM_Policy_Result01_Update02	The MDM_Policy_Result01_Update02 class represents the update policies available.
MDM_Policy_Result01_WiFi02	The MDM_Policy_Result01_WiFi02 class represents the Wi-Fi policies available.
MDM_Policy_Result01_WindowsInkWorkspace02	The MDM_Policy_Result01_WindowsInkWorkspace02 class represents the ink workspace policies available.
MDM_Policy_Result01_WirelessDisplay02	The MDM_Policy_Result01_WirelessDisplay02 class represents the wireless display policies available.
MDM_Policy_User_Config01_ApplicationManagement02	The MDM_Policy_User_Config01_ApplicationManagement02 class represents the start policies available that are set on a per-user basis.
MDM_Policy_User_Config01_Authentication02	The MDM_Policy_User_Config01_Authentication02 class represents the authentication management policies available.
MDM_Policy_User_Config01_Experience02	The MDM_Policy_User_Config01_Experience02 class represents the experience policies available.
MDM_Policy_User_Config01_Notifications02	The MDM_Policy_User_Config01_Notifications02 class represents the notification policies available.
MDM_Policy_User_Config01_Start02	The MDM_Policy_User_Config01_Start02 class represents the start policies available that are set on a per-user basis.
MDM_Policy_User_Result01_ApplicationManagement02	The MDM_Policy_User_Result01_ApplicationManagement02 class represents the start policies available that are set on a per-user basis.
MDM_Policy_User_Result01_Authentication02	The MDM_Policy_User_Result01_Authentication02 class represents the authentication management policies available.
MDM_Policy_User_Result01_Experience02	The MDM_Policy_User_Result01_Experience02 class represents the experience policies available.
MDM_Policy_User_Result01_Notifications02	The MDM_Policy_User_Result01_Notifications02 class represents the notification policies available.
MDM_Policy_User_Result01_Start02	The MDM_Policy_User_Result01_Start02 class represents the start policies available that are set on a per-user basis.
MDM_Reboot	The MDM_Rebootclass is used to configure reboot settings.
MDM_Reboot_Schedule01	The MDM_Reboot_Schedule01class is used to configure a specific time for the reboot of a device.

TOPIC	DESCRIPTION
MDM_RemoteFind	The MDM_RemoteFind class retrieves the location information for a particular device.
MDM_RemoteFind_Location01	The MDM_RemoteFind_Location01 class retrieves the location information for a particular device.
MDM_RemoteWipe	The MDM_RemoteWipe class allows a remote wipe of a device.
MDM_Reporting_EnterpriseDataProtection01_RetrieveByCount02	The MDM_Reporting_EnterpriseDataProtection01_RetrieveByCount02 class is used to retrieve a specified number of logs from the StartTime.
MDM_Reporting_EnterpriseDataProtection01_RetrieveByTimeRange02	The MDM_Reporting_EnterpriseDataProtection01_RetrieveByTimeRange02 class is used to retrieve the logs that exist within the StartTime and StopTime.
MDM_Reporting_SecurityAuditing01_RetrieveByCount02	The MDM_Reporting_SecurityAuditing01_RetrieveByCount02 class is used to retrieve a specified number of logs from the StartTime.
MDM_Reporting_SecurityAuditing01_RetrieveByTimeRange02	The MDM_Reporting_SecurityAuditing01_RetrieveByTimeRange02 class is used to retrieve the logs that exist within the StartTime and StopTime.
MDM_SecureAssessment	The MDM_SecureAssessmentclass is used to provide configuration information for the secure assessment browser.
MDM_Update	The MDM_Update class is used to manage and control the rollout of new updates.
MDM_Update_ApprovedUpdates01_01	The MDM_Update_ApprovedUpdates01_01 class is used to manage and control the rollout of approved updates.
MDM_Update_FailedUpdates01_01	The MDM_Update_FailedUpdates01_01 class is used to manage failed updates.
MDM_Update_InstallableUpdates01_01	The MDM_Update_InstallableUpdates01_01 class is used to manage and control the rollout of approved updates.
MDM_Update_PendingRebootUpdates01_01	The MDM_Update_PendingRebootUpdates01_01 class is used to manage updates that are pending on reboot.
MDM_VPNv2_01	The MDM_VPNv2_01 class allows configuration of the VPN profile of the device.

TOPIC	DESCRIPTION
MDM_VPNv2_APNBinding02	Reserved for Future Use
MDM_VPNv2_AppTriggerList02_App04	The MDM_VPNv2AppTriggerList02_App04 class describes a list of applications set to trigger the VPN.
MDM_VPNv2_Authentication03	The MDM_VPNv2_Authentication03 class contains authentication information for the native VPN profile.
MDM_VPNv2_Certificate04	Reserved for future Use
MDM_VPNv2_CryptographySuite03	The MDM_VPNv2_CryptographySuite03 class contains cryptography information for the native VPN profile.
MDM_VPNv2_DeviceCompliance02	Reserved for Future Use
MDM_VPNv2_DomainNameInformationList02_01	The MDM_VPNv2_DomainNameInformationList02_01 class describes the Name Resolution Policy Table (NRPT) rules for the VPN profile.
MDM_VPNv2_Eap04	The MDM_VPNv2_Eap04 class contains the required configuration XML information when the native profile specifies EAP authentication.
MDM_VPNv2_Manual03	The MDM_VPNv2_Manual03class is an optional node containing the manual server settings.
MDM_VPNv2_NativeProfile02	The MDM_VPNv2_NativeProfile2 class defines profile information when using a Windows Inbox VPN Protocol (IKEv2, PPTP, L2TP).
MDM_VPNv2_PluginProfile02	The MDM_VPNv2_PlugInProfile02 class describes the information needed when using a Windows Store based VPN plugin.
MDM_VPNv2_Proxy02	The MDM_VPNv2_Proxy02 class defines a configuration object to enable a post-connect proxy support for VPN.
MDM_VPNv2_RouteList02_01	The MDM_VPNv2_RouteList02_01 class contains an optional list of routes to be added to the routing table for the VPN interface.
MDM_VPNv2_Sso03	The MDM_VPNv2_Sso03 class can be used to select a certificate different from the VPN Authentication certificate for the Kerberos Authentication in the case of Device Compliance.
MDM_VPNv2_TrafficFilterList02_01	The MDM_VPNv2_TrafficFilterList02_01 class contains an optional list of rules. Only traffic that matches these rules can be sent via the VPN Interface.
MDM_VPNv2_TrafficFilterList02_App04	The MDM_TrafficFilterList02_App04 class provides configuration of the apps that are allowed over the VPN interface.

TOPIC	DESCRIPTION
MDM_WindowsAdvancedThreatProtection	The MDM_WindowsAdvancedThreatProtection class is used to onboard and offboard endpoints for Windows Defender Advanced Threat Protection (WDATP).
MDM_WindowsAdvancedThreatProtection_Configuration01	The MDM_WindowsAdvancedThreatProtection_Configuration01 class is used to determine the configuration of Windows Defender Advanced Threat Protection (WDATP) endpoints.
MDM_WindowsAdvancedThreatProtection_HealthState01	The MDM_WindowsAdvancedThreatProtection_HealthState01 class is used to determine the health status of Windows Defender Advanced Threat Protection (WDATP) endpoints.
MDM_WindowsLicensing	The MDM_WindowsLicensing class is designed for licensing related management scenarios.
MDM_WindowsLicensing_Subscriptions01_01	The MDM_WindowsLicensing_Subscriptions01_01 class is designed for subscription-related licensing management scenarios.
MDM_PassportForWork_ExcludeSecurityDevices03	The MDM_PassportForWork_ExcludeSecurityDevices03 class defines the Trusted Platform Modules allowed to use with Windows Hello for Business.
MDM_Policy_Config01_Games02	The MDM_Policy_Config01_Games02 class configures the advanced gaming services.
MDM_Policy_Config01_Location02	The MDM_Policy_Config01_Location02 class configures the location service of the device.
MDM_Policy_Config01_WindowsLogon02	The MDM_Policy_Config01_WindowsLogon02 class configures the lock screen and network user interface at logon.
MDM_Policy_Result01_Games02	The MDM_Policy_Result01_Games02 class gets the settings of the advanced gaming services.
MDM_Policy_Result01_Location02	The MDM_Policy_Result01_Location02 class gets the settings of the location service of the device.
MDM_Policy_Result01_WindowsLogon02	The MDM_Policy_Result01_WindowsLogon02 class gets the settings of the lock screen and network user interface at logon.
MDM_Policy_User_Config01_Settings02	The MDM_Policy_User_Config01_Settings02 class configures additional calendars in the taskbar.
MDM_Policy_User_Result01_Settings02	The MDM_Policy_User_Result01_Settings02 class gets the settings for additional calendars in the taskbar.
MDM_SharedPC	The MDM_SharedPC class is used to configure settings for shared PC usage.

TOPIC	DESCRIPTION
MDM_Policy_Config01_ActiveXControls02	This policy setting determines which ActiveX installation sites standard users in your organization can use to install ActiveX controls on their computers. When this setting is enabled, the administrator can create a list of approved Activex Install sites specified by host URL.
MDM_Policy_Config01_ApplicationDefaults02	The MDM_Policy_Config01_ApplicationDefaults02 class allows an administrator to set default file type and protocol associations. When set, default associations will be applied on sign-in to the PC.
MDM_Policy_Config01_AppVirtualization02	The MDM_Policy_Config01_AppVirtualization02 class configures the available app virtualization policies.
MDM_Policy_Config01_Autoplay02	The MDM_Policy_Config01_Autoplay02 class configures the available autoplay policies
MDM_Policy_Config01_CredentialProviders02	The MDM_Policy_Config01_CredentialProviders02 class configures the available credential provider policies.
MDM_Policy_Config01_CredentialsUI02	The MDM_Policy_Config01_CredentialsUI02 class configures the available credential provider policies.
MDM_Policy_Config01_DataUsage02	The MDM_Policy_Config01_DataUsage02 class configures the available data usage policies
MDM_Policy_Config01_DeviceInstallation02	The MDM_Policy_Config01_DeviceInstallation02 class configures the available device installation policies.
MDM_Policy_Config01_Display02	The MDM_Policy_Config01_Display02 class configures the display policies.
MDM_Policy_Config01_ErrorReporting02	The MDM_Policy_Config01_ErrorReporting02 class configures the error reporting policies.
MDM_Policy_Config01_EventLogService02	The MDM_Policy_Config01_EventLogService02 class configures the event log behavior.
MDM_Policy_Config01_InternetExplorer02	The MDM_Policy_Config01_InternetExplorer02 class configures the Internet Explorer policies.
MDM_Policy_Config01_Kerberos02	The MDM_Policy_Config01_Kerberos02 class configures the Kerberos policies.
MDM_Policy_Config01_Power02	The MDM_Policy_Config01_Power02 class configures the power policies.
MDM_Policy_Config01_Printers02	The MDM_Policy_Config01_Printers02 class configures the printer policies.
MDM_Policy_Config01_RemoteAssistance02	The MDM_Policy_Config01_RemoteAssistance02 class configures the remote assistance policies.

TOPIC	DESCRIPTION
MDM_Policy_Config01_RemoteDesktopServices02	The MDM_Policy_Config01_RemoteDesktopServices02 class configures the remote desktop service policies.
MDM_Policy_Config01_RemoteProcedureCall02	The MDM_Policy_Config01_RemoteProcedureCall02 class configures the remote procedure call policies.
MDM_Policy_Config01_SmartScreen02	The MDM_Policy_Config01_SmartScreen02 class configures the smart screen policies.
MDM_Policy_Config01_Storage02	The MDM_Policy_Config01_Storage02 class configures the storage policies.
MDM_Policy_Result01_ActiveXControls02	The MDM_Policy_Result01_ActiveXControls02 class represents the available ActiveX Controls policies.
MDM_Policy_Result01_ApplicationDefaults02	The MDM_Policy_Result01_ApplicationDefaults02 class represents the available application defaults policies.
MDM_Policy_Result01_AppVirtualization02	The MDM_Policy_Result01_AppVirtualization02 class represents the available app virtualization policies.
MDM_Policy_Result01_Autoplay02	The MDM_Policy_Result01_Autoplay02 class represents the available autoplay policies.
MDM_Policy_Result01_CredentialProviders02	The MDM_Policy_Result01_CredentialProviders02 class represents the available credential provider policies.
MDM_Policy_Result01_CredentialsUI02	The MDM_Policy_Result01_CredentialsUI02 class represents the available credentials policies.
MDM_Policy_Result01_DataUsage02	The MDM_Policy_Result01_DataUsage02 class represents the available data usage policies.
MDM_Policy_Result01_DeviceInstallation02	The MDM_Policy_Result01_DeviceInstallation02 class represents the available device installation policies.
MDM_Policy_Result01_Display02	The MDM_Policy_Result01_Display02 class represents the display policies.
MDM_Policy_Result01_ErrorReporting02	The MDM_Policy_Result01_ErrorReporting02 class represents the error reporting policies.
MDM_Policy_Result01_EventLogService02	The MDM_Policy_Result01_EventLogService02 class represents the event log behavior.
MDM_Policy_Result01_InternetExplorer02	The MDM_Policy_Result01_InternetExplorer02 represents the Internet Explorer policies.
MDM_Policy_Result01_Kerberos02	The MDM_Policy_Result01_Kerberos02 class represents the Kerberos policies.
MDM_Policy_Result01_Power02	The MDM_Policy_Result01_Power02 class represents the power policies.

TOPIC	DESCRIPTION
MDM_Policy_Result01_Printers02	The MDM_Policy_Result01_Printers02 class represents the printer policies.
MDM_Policy_Result01_RemoteAssistance02	The MDM_Policy_Result01_RemoteAssistance02 class represents the remote assistance policies.
MDM_Policy_Result01_RemoteDesktopServices02	The MDM_Policy_Result01_RemoteDesktopServices02 class represents the remote desktop services policies.
MDM_Policy_Result01_RemoteProcedureCall02	The MDM_Policy_Result01_RemoteProcedureCall02 class represents the remote procedure call policies.
MDM_Policy_Result01_SmartScreen02	the MDM_Policy_Result01_SmartScreen02 class represents the smart screen policies.
MDM_Policy_Result01_Storage02	The MDM_Policy_Result01_Storage02 class represents the available storage policies.
MDM_Policy_User_Config01_AttachmentManager02	The MDM_Policy_User_Config01_AttachmentManager02 class represents the available attachment manager policies.
MDM_Policy_User_Config01_CredentialsUI02	The MDM_Policy_User_Config01_CredentialsUI02 class represents the available credentials policies.
MDM_Policy_User_Config01_Desktop02	The MDM_Policy_User_Config01_Desktop02 class represents the available folder profile policies.
MDM_Policy_User_Config01_EnterpriseCloudPrint02	The MDM_Policy_User_Config01_EnterpriseCloudPrint02 class represents the available cloud print policies.
MDM_Policy_User_Config01_InternetExplorer02	The MDM_Policy_User_Config01_InternetExplorer02 class represents the available Internet Explorer policies.
MDM_Policy_User_Config01_Printers02	The MDM_Policy_User_Config01_Printers02 class represents the available printer policies.
MDM_Policy_User_Config01_System02	The MDM_Policy_User_Config01_System02 class represents the available telemetry policies.
MDM_Policy_User_Result01_AttachmentManager02	The MDM_Policy_User_Result01_AttachmentManager02 class represents the available attachment manager policies.
MDM_Policy_User_Result01_CredentialsUI02	The MDM_Policy_User_Result01_CredentialsUI02 class represents the available credentials policies.
MDM_Policy_User_Result01_Desktop02	The MDM_Policy_User_Result01_Desktop02 class represents the available folder profile policies.
MDM_Policy_User_Result01_EnterpriseCloudPrint02	The MDM_Policy_User_Result01_EnterpriseCloudPrint02 class represents the available cloud print policies.
MDM_Policy_User_Result01_InternetExplorer02	The MDM_Policy_User_Result01_InternetExplorer02 class represents the available Internet Explorer policies.

TOPIC	DESCRIPTION
MDM_Policy_User_Result01_Printers02	The MDM_Policy_User_Result01_Printers02 class represents the available printer policies.
MDM_Policy_User_Result01_System02	The MDM_Policy_User_Result01_System02 class represents the available telemetry policies.
MDM_Policy_Config01_Cellular02	The MDM_Policy_Config01_Cellular02 class configures the cellular policies.
MDM_Policy_Config01_DeviceGuard02	The MDM_Policy_Config01_DeviceGuard02 configures the Device Guard policies.
MDM_Policy_Config01_LocalPoliciesSecurityOptions02	The MDM_Policy_Config01_LocalPoliciesSecurityOptions02 configures the local policies security options.
MDM_Policy_Config01_RemoteManagement02	The MDM_Policy_Config01_RemoteManagement02 remote management policies.
MDM_Policy_Config01_RemoteShell02	The MDM_Policy_Config01_RemoteShell02 class configures the remote shell policies.
MDM_Policy_Config01_WindowsDefenderSecurityCenter02	The MDM_Policy_Config01_WindowsDefenderSecurityCenter02 class represents the Windows Defender Security Center policies.
MDM_Policy_Result01_Cellular02	the MDM_Policy_Result01_Cellular02 class represents the cellular policies.
MDM_Policy_Result01_DeviceGuard02	The MDM_Policy_Result01_DeviceGuard02 class configures the Device Guard policies.
MDM_Policy_Result01_LocalPoliciesSecurityOptions02	The MDM_Policy_Result01_LocalPoliciesSecurityOptions02 class represents the local policies security options.
MDM_Policy_Result01_RemoteManagement02	The MDM_Policy_Result01_RemoteManagement02 class represents the remote management policies.
MDM_Policy_Result01_RemoteShell02	The MDM_Policy_Result01_RemoteShell02 class represents the remote shell policies.
MDM_Policy_Result01_WindowsDefenderSecurityCenter02	The MDM_Policy_Result01_WindowsDefenderSecurityCenter02 class represents the Windows Defender Security Center policies.
MDM_Policy_User_Config01_Education02	The MDM_Policy_User_Config01_Education02 class configures the education policies.
MDM_Policy_User_Result01_Education02	The MDM_Policy_User_Result01_Education02 class represents the education policies.
MDM_BitLocker	The MDM_BitLocker class is used by the enterprise to manage encryption of PCs and devices.

TOPIC	DESCRIPTION
MDM_DeviceManageability_Provider01_01	The MDM_DeviceManageability_Provider01_01 class is used retrieve the general information about MDM configuration capabilities on the device.
MDM_Firewall_Action04	The MDM_Firewall_Action04 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_App04	The MDM_Firewall_App04 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_DomainProfile02	The MDM_Firewall_DomainProfile02 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_FirewallRules02_01	The MDM_Firewall_FirewallRules02_01 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_Global02	The MDM_Firewall_Global02 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_PrivateProfile02	The MDM_Firewall_PrivateProfile02 class is used to configure the Windows Defender Firewall settings.
MDM_Firewall_PublicProfile02	The MDM_Firewall_PublicProfile02 class is used to configure the Windows Defender Firewall settings.
MDM_Personalization	The MDM_Personalization class is used to set the lock screen and desktop background images. Setting these policies also prevents the user from changing the image.
MDM_WindowsAdvancedThreatProtection_DeviceTagging01	The MDM_WindowsAdvancedThreatProtection_DeviceTagging01 class is used to onboard, determine configuration and health status, and offboard endpoints for Windows Defender Advanced Threat Protection.
MDM_DeviceStatus_DeviceGuard01	The MDM_DeviceStatus_DeviceGuard01 class is used by the enterprise to keep track of device inventory and query the state of compliance of these devices with their enterprise policies.
MDM_Policy_Config01_ExploitGuard02	The MDM_Policy_Config01_ExploitGuard02 is used to push out a configuration representing the desired system and application mitigation options in Windows Defender Exploit Guard to all the devices in the organization. The configuration is represented by an XML. For more information Exploit Protection, see Protect devices from exploits with Windows Defender Exploit Guard and IImport, export, and deploy Exploit Protection configurations.
MDM_Policy_Config01_Handwriting02	The MDM_Policy_Config01_Handwriting02 class is used to configure the default mode for the handwriting panel.

TOPIC	DESCRIPTION
MDM_Policy_Config01_Messaging02	The MDM_Policy_Config01_Messaging02 class enables text message back up and restore and Messaging Everywhere. This policy allows an organization to disable these features to avoid information being stored on servers outside of their control.
MDM_Policy_Result01_ExploitGuard02	The MDM_Policy_Result01_ExploitGuard02 class represents the configuration setting in Windows Defender Exploit Guard. The configuration is represented by an XML. For more information Exploit Protection, see Protect devices from exploits with Windows Defender Exploit Guard and IImport, export, and deploy Exploit Protection configurations.
MDM_Policy_Result01_Handwriting02	The MDM_Policy_Result01_Handwriting02 class represents the default mode for the handwriting panel.
MDM_Policy_Result01_Messaging02	The MDM_Policy_Result01_Messaging02 class represents the text message back up and restore and Messaging Everywhere.

Windows System Assessment Tool Provider

2/22/2020 • 2 minutes to read • Edit Online

[WinSAT may be altered or unavailable for releases after Windows 8.1.]

Purpose

The Windows System Assessment Tool (WinSAT) exposes a number of classes that assesses the performance characteristics and capabilities of a computer. Developers can use this API to develop software that can access the performance and capability information of a computer to determine the optimal application settings based on that computer's performance capabilities.

Users can use the results of a formal assessment to determine which scenarios and applications will perform well on a computer. For example, if a software package contains a rating on its packaging, a user can use the rating to determine whether the software will run well on the computer.

Developer audience

WinSAT is designed for C, C++, and Visual Basic developers and those who write scripts.

Run-time requirements

WinSAT is available on client computers starting with the Windows Vista operating system.

In this section

TOPIC	DESCRIPTION
Using WinSAT	Procedural guide for initiating assessments, retrieving scores, and retrieving assessment details.
WinSAT Reference	Reference information for the WinSAT API, samples, and schema.

WMI Core Provider

2/18/2021 • 2 minutes to read • Edit Online

The WMI Core provider defines classes that compose the core functionality of WMI.

A majority of the WMI Core provider classes contain data supplied by the WDM Provider, and describe display monitors. The classes are defined in Wmi.mof and are located in the "Root\WMI" namespace.

In this section

TOPIC	DESCRIPTION
MSMonitorClass	is an abstract WMI base class. The classes that describe video display monitors inherit from this MSMonitorClass.
MSMCA Classes	a set of WMI classes that expose the Machine Check Architecture (MCA). The System Abstraction Layer (SAL) provides all events reported in the MSMCA class. The Intel Corporation develops and owns the MCA.
FrequencyRangeDescriptor	represents a container for characteristics of a supported frequency range.
SupportedDisplayFeaturesDescriptor	represents the supported display features of the monitor.
VideoModeDescriptor	contains mode descriptor elements for the **MonitorSourceModes** array in the WmiMonitorListedSupportedSourceModes class.
WMIEvent	The WMIEvent class is a base class from which all WMI event classes are derived.
WmiMonitorAnalogVideoInputParams	represents the analog video input parameters of a computer monitor.
WmiMonitorBasicDisplayParams	represents the basic display features of a computer monitor.
WmiMonitorBrightness	represents the brightness parameters of a computer monitor.
WmiMonitorBrightnessEvent	represents a change in the brightness of a monitor.
WmiMonitorBrightnessMethods	contains methods that manage monitor brightness.
WmiMonitorColorCharacteristics	represents the International Commission on Illumination (CIE) color characteristics of a computer monitor.
WmiMonitorConnectionParams	contains the connection type of the monitor.

TOPIC	DESCRIPTION
WmiMonitorDescriptorMethods	contains methods that obtain the raw content of Video Input Definition of Video Electronics Standard Association (VESA) Enhanced Extended Display Identification Data (E-EDID) v.1.x standard 128-byte data blocks.
WmiMonitorDigitalVideoInputParams	represents input parameters for digital video.
WmiMonitorID	represents the identifying information about a video monitor, such as manufacturer name, year of manufacture, or serial number.
WmiMonitorListedFrequencyRanges	lists the frequency ranges supported by the monitor.
WmiMonitorListedSupportedSourceModes	lists the supported source modes for a video monitor in its monitor descriptor, if any exist.
WmiMonitorRawEEdidV1Block	represents the raw data from a Video Electronics Standard Association (VESA) Enhanced Extended Display Identification Data (E-EDID) structure.
XYZinCIE	contains the coordinates of the display in the International Commission on Illumination (CIE) XYZ color space.

Windows Remote Management

11/2/2020 • 2 minutes to read • Edit Online

Purpose

Windows Remote Management (WinRM) is the Microsoft implementation of WS-Management Protocol, a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate.

The WS-Management protocol specification provides a common way for systems to access and exchange management information across an IT infrastructure. WinRM and *Intelligent Platform Management Interface (IPMI)*, along with the Event Collector are components of the Windows Hardware Management features.

Where applicable

You can use WinRM scripting objects, the WinRM command-line tool, or the Windows Remote Shell command line tool WinRS to obtain management data from local and remote computers that may have *baseboard management controllers (BMCs)*. If the computer runs a Windows-based operating system version that includes WinRM, the management data is supplied by Windows Management Instrumentation (WMI).

You can also obtain hardware and system data from WS-Management protocol implementations running on operating systems other than Windows in your enterprise. WinRM establishes a session with a remote computer through the SOAP-based WS-Management protocol rather than a connection through DCOM, as WMI does. Data returned to WS-Management protocol are formatted in XML rather than in objects.

The Intelligent Platform Management Interface (IPMI) WMI provider is a standard WMI provider with classes that obtain BMC sensor data from computers with appropriate hardware. IPMI data can be accessed using the WinRM scripting API, the WMI Scripting, or COM APIs.

Developer audience

The developer audience is the IT Pro who writes scripts to automate the management of servers or the ISV developer obtaining data for management applications.

Run-time requirements

WinRM is part of the operating system. However, to obtain data from remote computers, you must configure a WinRM *listener*. For more information, see Installation and Configuration for Windows Remote Management. If a BMC is detected at system startup, then the IPMI provider loads; otherwise, the WinRM scripting objects and the WinRM command-line tool are still available.

In this section

About Windows Remote Management

Link to public WS-Management protocol specification, WinRM architecture, relationship to WMI, hardware management with the IPMI provider, configuration and installation.

Using Windows Remote Management

Getting started using the WinRM scripting API and hardware management.

Windows Remote Management Reference

List of scripting interfaces defined by Microsoft Web Services for Management (WS-Management) Automation and class definitions of the WMI classes created by the IPMI provider and classes that communicate with the IPMI driver to obtain baseboard management controller (BMC) data.

Windows Resource Protection

2/18/2021 • 2 minutes to read • Edit Online

Purpose

Windows Resource Protection (WRP) prevents the replacement of essential system files, folders, and registry keys that are installed as part of the operating system. It became available starting with Windows Server 2008 and Windows Vista. Applications should not overwrite these resources because they are used by the system and other applications. Protecting these resources prevents application and operating system failures. WRP is the new name for Windows File Protection (WFP). WRP protects registry keys and folders as well as essential system files.

Where applicable

All Windows-based applications and their installation programs that run on Windows Server 2008 or Windows Vista, and later operating systems, should be aware of WRP. All Windows-based applications that run on Microsoft Windows Server 2003 or Windows XP and their installation programs should be aware of WFP.

Developer audience

The WRP API is designed for use by C/C++ programmers. The WRP API has two functions: **SfcIsFileProtected** and **SfcIsKeyProtected**.

Run-time requirements

Applications that use only the **SfcIsFileProtected** function require Windows Server 2008, Windows Vista, Windows Server 2003, or Windows XP. Applications that use the **SfcIsKeyProtected** function require Windows Server 2008 or Windows Vista.

In this section

TOPIC	DESCRIPTION
About Windows Resource Protection	General information about WRP.
Windows Resource Protection Reference	Reference documentation for WRP.

API Index for desktop Windows applications

11/2/2020 • 2 minutes to read • Edit Online

This article provides links to reference documentation for APIs that can be used in desktop Windows apps.

Win32 (Windows API)

The Win32 API (also called the Windows API) is the native platform for Windows apps. This API is best for desktop apps that require direct access to system features and hardware. The Windows API can be used in all desktop apps, and the same functions are generally supported on 32-bit and 64-bit Windows.

- Win32 API reference by feature
- Win32 API reference by header
- Win32 and COM APIs for UWP apps
- Windows umbrella libraries
- Windows API Sets

Windows Runtime (WinRT)

WinRT is the leading edge platform for Windows 10 apps and games, including desktop apps. The WinRT API is suitable for both native C++ and managed desktop apps that require a sophisticated UI, styles customization, and graphics-intensive scenarios.

- WinRT API reference
- WinRT APIs callable from Win32, WPF, and Windows Forms desktop apps

.NET

The .NET class libraries provide access to Windows system and UI features for managed desktop apps, including WPF and Windows Forms apps.

- .NET API

Windows Runtime C++ reference

2/18/2021 • 2 minutes to read • Edit Online

In this section

- Data types
- Enumerations
- Functions
- Interfaces
- Structures

Certify your desktop application

11/2/2020 • 2 minutes to read • Edit Online

IMPORTANT

Certification for Win32 desktop apps is deprecated. Instead, submit files here.

Follow these steps to get your desktop app certified for Windows 7, Windows 8, Windows 8.1, and Windows 10.

If you wish to convert your desktop app to be compatible with the Universal Windows Platform and Windows Store, you will use the Windows Desktop Bridge, in which case you should follow the Desktop Bridge guidance for certification steps.

If you are developing and certifying a UWP app from the start, see Guidance for Windows App Certification in UWP for info on certification.

Step 1: Prepare for certification

TOPIC	DESCRIPTION
What are the benefits?	Certifying your desktop app provides several benefits for you and your customers.
Read the requirements	Review the technical requirements and eligibility qualifications that a desktop app must meet.

Step 2: Test your app with the Windows App Certification Kit

TOPIC	DESCRIPTION
Get the Kit	To certify your app, you have to install and run the Windows App Certification Kit (included in the Windows SDK).
Using the Kit	Before you can submit your app, you must test it for readiness. You can also download a copy of the app certification white paper.
Review test details	Get the list of the tests your app needs to pass to qualify for compatibility with the latest Windows operating system.

Note: Filter drivers must also pass the Hardware Certification Kit. (See Certification requirements for Windows desktop apps, section 6.2.)

Step 3: Use the Windows Certification Dashboard

To submit your app for certification, you'll need to log in or register a company account on the Windows Certification Dashboard. Once you do, not only will you be able to get your app certified, but you'll also gain access to tools to review and manage your app at all stages of the process.

TOPIC	DESCRIPTION
Set up your account	If your company isn't already registered, you must register it through the Windows Certification Dashboard.
Get a code signing certificate	Before you can establish a Windows Certification Dashboard account, you need to get a code signing certificate to secure your digital information.
Test locally and upload the results	After your run the Windows App Certification Kit tests, upload the results to the Windows Certification Dashboard.
Manage your submission	After you submit your app for certification, you can review your submission through the Windows Certification Dashboard.

Step 4: Promote your desktop app

TOPIC	DESCRIPTION
Check app compatibility	If you are building an app for Windows 8.1, investigate compatibility concerns.
Use the logo with your app	Display the logo on packaging and in ads and other promotional materials according to the guidelines. For Windows 7 only.

See also:

App compatibility forum: Find support from the community about compatibility and logo certification.

Windows SDK blog: Find tips and news related to app certification.

Windows Server forum: Visit the Certification forum to get answers.

Compatibility cookbook: Get info about what's new or changed in the latest version of Windows.