

.NET Framework documentation

Learn about .NET Framework, a development platform for building apps for web, Windows, and Microsoft Azure.

Learn about .NET Framework

OVERVIEW

[.NET Framework overview](#)

[System requirements](#)

GET STARTED

[.NET Framework on Q&A](#)

CONCEPT

[Older versions of .NET Framework](#)

DEPLOY

[Deployment guide for developers](#)

REFERENCE

[.NET Framework APIs](#)

Install .NET Framework

DOWNLOAD

[Download the .NET SDK ↗](#)

OVERVIEW

[Installation guide](#)

CONCEPT

.NET Framework & Windows versions

HOW-TO GUIDE

[Determine which versions are installed](#)

[Install .NET Framework on Windows 10](#)

[Install .NET Framework 3.5](#)

[Install the developer pack or redistributable](#)

[Troubleshoot blocked installations](#)

Create Windows service apps

OVERVIEW

[Introduction to Windows service apps](#)

TUTORIAL

[Create a Windows service app](#)

HOW-TO GUIDE

[Install and uninstall Windows services](#)

Create desktop apps

OVERVIEW

[Desktop guide Introduction](#)

[.NET Framework WPF](#)

[.NET Framework Windows Forms](#)

Create WPF apps

OVERVIEW

Overview of WPF

TUTORIAL

[Create your first WPF app in Visual Studio](#)

Work with data using ADO.NET

OVERVIEW

[Overview of ADO.NET](#)

TUTORIAL

[Model-first development using Entity Framework](#)

[Database-first development using Entity Framework](#)

CONCEPT

[Connection strings and configuration files](#)

[SQL Server connection pooling](#)

HOW-TO GUIDE

[Retrieve data using a DataReader](#)

[Add data to a DataTable](#)

REFERENCE

[Connection string syntax](#)

[Sample SQL Server databases](#)

Overview of .NET Framework

Article • 03/30/2023

.NET Framework is a technology that supports building and running Windows apps and web services. .NET Framework is designed to fulfill the following objectives:

- Provide a consistent, object-oriented programming environment whether object code is stored and executed locally, executed locally but web-distributed, or executed remotely.
- Provide a code-execution environment that:
 - Minimizes software deployment and versioning conflicts.
 - Promotes safe execution of code, including code created by an unknown or semi-trusted third party.
 - Eliminates the performance problems of scripted or interpreted environments.
- Make the developer experience consistent across widely varying types of apps, such as Windows-based apps and Web-based apps.
- Build all communication on industry standards to ensure that code based on .NET Framework integrates with any other code.

ⓘ Note

.NET Framework is serviced monthly  with security and reliability bug fixes. .NET Framework will continue to be included with Windows, with no plans to remove it. You don't need to migrate your .NET Framework apps, but for new development, use [.NET 6 or later](#).

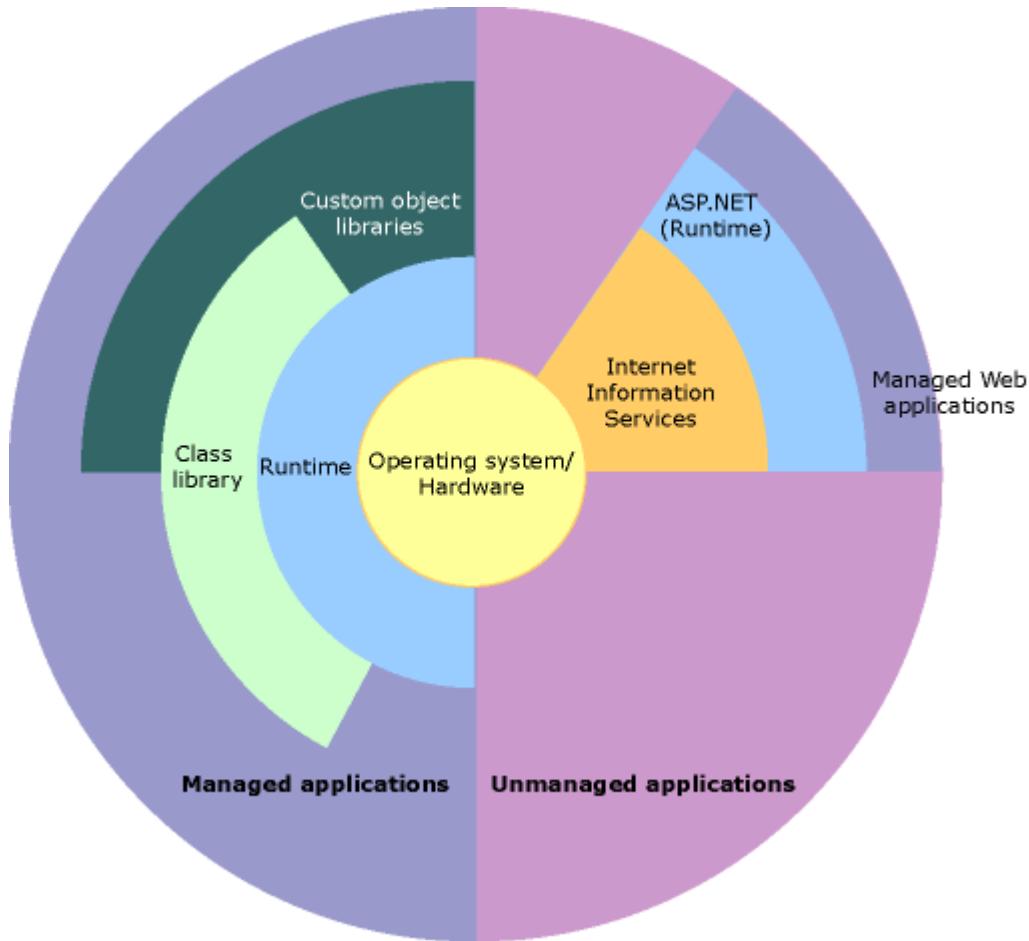
.NET Framework consists of the common language runtime (CLR) and the .NET Framework class library. The common language runtime is the foundation of .NET Framework. Think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and remoting, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that doesn't target the runtime is known as unmanaged code. The class library is a comprehensive, object-oriented collection of reusable types that you use to develop apps ranging from traditional command-line or graphical user

interface (GUI) apps to apps based on the latest innovations provided by ASP.NET, such as Web Forms and XML web services.

.NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that exploits both managed and unmanaged features. .NET Framework not only provides several runtime hosts but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable ASP.NET apps and XML web services, both of which are discussed later in this article.

The following illustration shows the relationship of the common language runtime and the class library to your apps and to the overall system. The illustration also shows how managed code operates within a larger architecture.



The following sections describe the main features of .NET Framework in greater detail.

Features of the common language runtime

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are

intrinsic to the managed code that runs on the common language runtime.

Regarding security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it's used in the same active app.

The runtime also enforces code robustness by implementing a strict type-and-code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common app errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers write apps in their development language of choice yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing apps.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it's executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side apps, such as Microsoft SQL Server and Internet Information Services (IIS). This infrastructure enables

you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

.NET Framework class library

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code derives functionality. This not only makes the .NET Framework types easy to use but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces for developing your own collection classes. Your collection classes blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. You can use .NET Framework to develop the following types of apps and services:

- Console apps. See [Building Console Applications](#).
- Windows GUI apps (Windows Forms). See [Windows Forms](#).
- Windows Presentation Foundation (WPF) apps. See [Windows Presentation Foundation](#).
- ASP.NET apps. See [Web Applications with ASP.NET](#).
- Windows services. See [Introduction to Windows Service Applications](#).
- Service-oriented apps using Windows Communication Foundation (WCF). See [Service-Oriented Applications with WCF](#).
- Workflow-enabled apps using Windows Workflow Foundation (WF). See [Windows Workflow Foundation](#).

The Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form app, you can use the Web Forms classes.

See also

- [System Requirements](#)
- [Installation guide](#)
- [Development guide](#)
- [Tools](#)
- [.NET samples and tutorials](#)
- [.NET API browser](#)

Get started with .NET Framework

Article • 09/21/2022

.NET Framework is a run-time execution environment that manages apps that target .NET Framework. It consists of the common language runtime, which provides memory management and other system services, and an extensive class library, which enables programmers to take advantage of robust, reliable code for all major areas of app development.

ⓘ Note

.NET Framework [is serviced monthly](#) with security and reliability bug fixes. .NET Framework will continue to be included with Windows, with no plans to remove it. You don't need to migrate your .NET Framework apps, but for new development, use [.NET 6 or later](#).

What is .NET Framework?

.NET Framework is a managed execution environment for Windows that provides a variety of services to its running apps. It consists of two major components: the common language runtime (CLR), which is the execution engine that handles running apps, and the .NET Framework Class Library, which provides a library of tested, reusable code that developers can call from their own apps. The services that .NET Framework provides to running apps include the following:

- Memory management. In many programming languages, programmers are responsible for allocating and releasing memory and for handling object lifetimes. In .NET Framework apps, the CLR provides these services on behalf of the app.
- A common type system. In traditional programming languages, basic types are defined by the compiler, which complicates cross-language interoperability. In .NET Framework, basic types are defined by the .NET Framework type system and are common to all languages that target .NET Framework.
- An extensive class library. Instead of having to write vast amounts of code to handle common low-level programming operations, programmers use a readily accessible library of types and their members from the .NET Framework Class Library.

- Development frameworks and technologies. .NET Framework includes libraries for specific areas of app development, such as ASP.NET for web apps, ADO.NET for data access, Windows Communication Foundation for service-oriented apps, and Windows Presentation Foundation for Windows desktop apps.
- Language interoperability. Language compilers that target .NET Framework emit an intermediate code named Common Intermediate Language (CIL), which, in turn, is compiled at run time by the common language runtime. With this feature, routines written in one language are accessible to other languages, and programmers focus on creating apps in their preferred languages.
- Version compatibility. With rare exceptions, apps that are developed by using a particular version of .NET Framework run without modification on a later version.
- Side-by-side execution. .NET Framework helps resolve version conflicts by allowing multiple versions of the common language runtime to exist on the same computer. This means that multiple versions of apps can coexist and that an app can run on the version of .NET Framework with which it was built. Side-by-side execution applies to the .NET Framework version groups 1.0/1.1, 2.0/3.0/3.5, and 4/4.5.x/4.6.x/4.7.x/4.8.x.
- Multitargeting. By targeting [.NET Standard](#), developers create class libraries that work on multiple .NET Framework platforms supported by that version of the standard. For example, libraries that target .NET Standard 2.0 can be used by apps that target .NET Framework 4.6.1, .NET Core 2.0, and UWP 10.0.16299.

.NET Framework for users

If you don't develop .NET Framework apps, but you use them, you aren't required to have specific knowledge about .NET Framework or its operation. For the most part, the framework is completely transparent to users.

If you're using the Windows operating system, .NET Framework may already be installed on your computer. In addition, if you install an app that requires .NET Framework, the app's setup program might install a specific version of the framework on your computer. In some cases, you may see a dialog box that asks you to install .NET Framework. If you've just tried to run an app when this dialog box appears and if your computer has internet access, you can go to a webpage that lets you install the missing version of .NET Framework. For more information, see the [Installation guide](#).

In general, you shouldn't uninstall versions of .NET Framework that are installed on your computer. There are two reasons for this:

- If an app that you use depends on a specific version of .NET Framework, that app may break if that version is removed.
- Some versions of .NET Framework are in-place updates to earlier versions. For example, .NET Framework 3.5 is an in-place update to version 2.0, and .NET Framework 4.8 is an in-place update to versions 4 through 4.7.2. For more information, see [.NET Framework Versions and Dependencies](#).

On Windows versions before Windows 8, if you do choose to remove .NET Framework, always use **Programs and Features** from Control Panel to uninstall it. Never remove a version of .NET Framework manually. On Windows 8 and above, .NET Framework is an operating system component and cannot be independently uninstalled.

Multiple versions of .NET Framework can coexist on a single computer at the same time. This means that you don't have to uninstall previous versions in order to install a later version.

.NET Framework for developers

If you're a developer, choose any programming language that supports .NET Framework to create your apps. Because .NET Framework provides language independence and interoperability, you interact with other .NET Framework apps and components regardless of the language with which they were developed.

To develop .NET Framework apps or components, do the following:

1. If it's not preinstalled on your operating system, install the version of .NET Framework that your app will target. The current versions are .NET Framework 4.8 and .NET Framework 4.8.1. .NET Framework 4.8.1 is [available for download](#) on the latest versions of Windows and Windows Server. .NET Framework 4.8 is preinstalled on Windows 10 May 2019 Update, Windows 10 November 2019 Update, Windows 10 May 2020 Update, and Windows 10 October 2020 Update, and it's available for download on earlier versions of the Windows operating system. For .NET Framework system requirements, see [System Requirements](#). For information on installing other versions of .NET Framework, see [Installation Guide](#). Additional .NET Framework packages are released out of band, which means that they're released on a rolling basis outside of any regular or scheduled release cycle. For information about these packages, see [.NET Framework and Out-of-Band Releases](#).
2. Select the language or languages supported by the .NET Framework version that you intend to use to develop your apps. A number of languages are available, including [Visual Basic](#), [C#](#), [F#](#), and [C++/CLI](#) from Microsoft. (A programming

language that allows you to develop apps for .NET Framework adheres to the [Common Language Infrastructure \(CLI\) specification](#).)

3. Select and install the development environment to use to create your apps and that supports your selected programming language or languages. The Microsoft integrated development environment (IDE) for .NET Framework apps is [Visual Studio](#). It's available in a number of editions.

For more information on developing apps that target .NET Framework, see the [Development Guide](#).

Related articles

[] [Expand table](#)

Title	Description
Overview	Provides detailed information for developers who build apps that target .NET Framework.
Installation guide	Provides information about installing .NET Framework.
.NET Framework and Out-of-Band Releases	Describes the .NET Framework out-of-band releases and how to use them in your app.
System Requirements	Lists the hardware and software requirements for running .NET Framework.
.NET Core documentation	Provides the conceptual and API reference documentation for .NET Core.
.NET Standard	Discusses .NET Standard, a versioned specification that individual .NET implementations support to guarantee that a consistent set of APIs is available on multiple platforms.

See also

- [.NET Framework guide](#)
- [What's new](#)
- [.NET API browser](#)
- [Development guide](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

.NET Framework and out-of-band releases

Article • 06/04/2024

.NET Framework has evolved to accommodate different platforms, such as UWP apps and traditional desktop and web apps, and to maximize code reuse. In addition to regular .NET Framework releases, new features are released out of band (OOB) to improve cross-platform development or to introduce new functionality.

Advantages of OOB releases

Shipping new components or updates to components out of band enables Microsoft to provide more frequent updates to .NET Framework. In addition, we can gather and respond to customer feedback more quickly.

When you use an OOB feature in your app, your users do not have to install the latest version of .NET Framework to run your app, because the OOB assemblies deploy with your app package.

How OOB packages are distributed

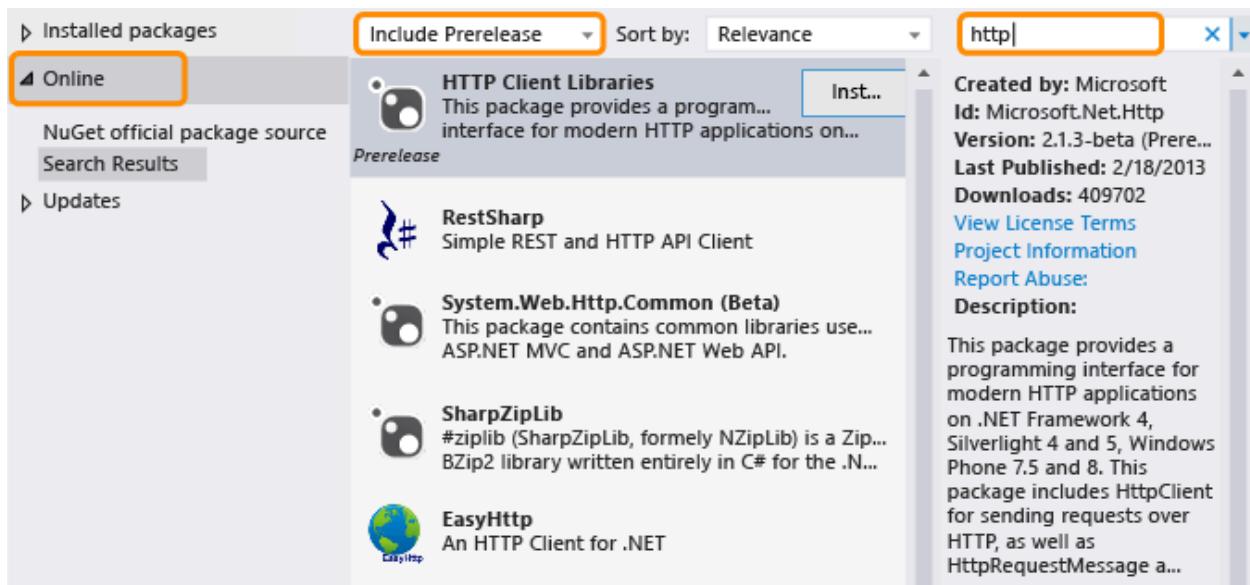
OOB releases for core common language runtime (CLR) components are delivered through [NuGet](#), which is the package manager for .NET. NuGet enables you to browse and add libraries to your .NET Framework projects easily from within Visual Studio. NuGet Package Manager is included with all editions of Visual Studio starting with Visual Studio 2012. Look for **NuGet Package Manager** on the **Tools** menu in Visual Studio. If it's not installed, follow the instructions on [Installing NuGet](#). For more information about NuGet, see the [NuGet docs](#).

Use a NuGet OOB package

If NuGet Package Manager is installed, you can browse and add references to NuGet packages by using Solution Explorer in Visual Studio:

1. Open the shortcut menu for your project in Visual Studio, and then choose **Manage NuGet Packages**. (This option is also available from the **Project** menu.)
2. In the left pane, choose **Online**.

3. If you want to use prerelease packages, in the drop-down list box in the middle pane, choose **Include Prerelease** instead of **Stable Only**.
4. In the right pane, use the **Search** box to locate the package you would like to use. Some Microsoft packages are identified by the Microsoft .NET Framework logo, and all identify Microsoft as the publisher.



As mentioned previously, when you deploy an app that uses an OOB package, the OOB assemblies will ship with your app package.

Types of OOB releases

Typically, an OOB package has one or more prerelease versions and a stable version. The license that accompanies a prerelease doesn't typically allow redistribution, but enables you to try out a package and provide feedback. Feedback is incorporated in any updates made to the package. A final release is distributed as a stable package with NuGet and includes a license that lets you redistribute the NuGet package with your app. Stable packages are supported by Microsoft. Microsoft provides IntelliSense support as well as other types of documentation such as blog posts and forum answers for all packages. In addition, source code may be available with some, but not all, packages. For announcements regarding new and updated packages, you can subscribe to [the .NET Framework Blog](#).

To find both prerelease and stable packages, choose **Include Prerelease** in NuGet Package Manager.

See also

- [Getting started](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

.NET Framework system requirements

Article • 04/26/2024

The tables in this article provide the hardware, operating system, and software requirements for the following .NET Framework versions:

- .NET Framework 4.8 and its point release (4.8.1).
- .NET Framework 4.7 and its point releases (4.7.1 and 4.7.2).
- .NET Framework 4.6.2.

For information on earlier .NET Framework versions, see [.NET Framework versions and dependencies](#).

Development environments that enable you to develop apps for .NET Framework have a separate set of requirements.

Important

All .NET Framework versions since .NET Framework 4 are in-place updates, so only a single 4.x version can be present on a system. In addition, particular versions of .NET Framework are preinstalled on some versions of the Windows operating system. This means that:

- If there's a later 4.x version installed on the machine already, you can't install a previous 4.x version.
- If the OS comes preinstalled with a particular .NET Framework version, you can't install a previous 4.x version on the same machine.
- If you install a later version, you don't have to first uninstall the previous version.

For download information and links, see [Install .NET Framework for developers](#).

For information on the support lifecycle of .NET Framework versions, see [Microsoft support lifecycle](#).

Hardware requirements

 Expand table

Requirement	
Processor	1 GHz
RAM	512 MB
Minimum disk space (32-bit)	4.5 GB
Minimum disk space (64-bit)	4.5 GB

Installation requirements

.NET Framework requires administrator privileges for installation. If you don't have administrator rights to the computer where you'd like to install .NET Framework, contact your network administrator.

Compatible operating systems

- [Client operating systems](#)
- [Server operating systems](#)

For all platforms, to ensure the best compatibility and security, we recommend that you install critical updates available from [Windows Update](#).

On 64-bit operating systems, .NET Framework supports both WOW64 (32-bit processing on a 64-bit machine) and native 64-bit processing.

Client operating systems

[\[+\] Expand table](#)

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
Windows 11, 2023 Update (version 23H2)	64-bit	.NET Framework 4.8.1	--
Windows 11, 2022 Update (version 22H2)	64-bit	.NET Framework 4.8.1	--
Windows 11	64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 2022 Update (version 22H2)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
Windows 10 November 2021 Update (version 21H2)	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 May 2021 Update (version 21H1)†	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 October 2020 Update (version 20H2)†	32-bit and 64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows 10 May 2020 Update (version 2004)†	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 November 2019 Update (version 1909)†	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 May 2019 Update (version 1903)†	32-bit and 64-bit	.NET Framework 4.8	--
Windows 10 October 2018 Update (version 1809)†	32-bit and 64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows 10 April 2018 Update (version 1803)†	32-bit and 64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows 10 Fall Creators Update (version 1709)†	32-bit and 64-bit	.NET Framework 4.7.1	.NET Framework 4.7.2 .NET Framework 4.8
Windows 10 Creators Update (version 1703)†	32-bit and 64-bit	.NET Framework 4.7	.NET Framework 4.7.1 .NET Framework 4.7.2 .NET Framework 4.8

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
Windows 10 Anniversary Update (version 1607)†	32-bit and 64-bit	.NET Framework 4.6.2	.NET Framework 4.7 .NET Framework 4.7.1 .NET Framework 4.7.2 .NET Framework 4.8
Windows 10 November Update (version 1511)†	32-bit and 64-bit	.NET Framework 4.6.1	.NET Framework 4.6.2
Windows 10 (version 1507)†	32-bit and 64-bit	.NET Framework 4.6	.NET Framework 4.6.2
Windows 8.1†	32-bit, 64-bit, and ARM	.NET Framework 4.5.1	.NET Framework 4.6.2 .NET Framework 4.7 .NET Framework 4.7.1 .NET Framework 4.7.2 .NET Framework 4.8
Windows 7 SP1†	32-bit and 64-bit	--	.NET Framework 4.6.2 .NET Framework 4.7 .NET Framework 4.7.1 .NET Framework 4.7.2

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
			.NET Framework 4.8

[†]The following operating systems, while listed in the table, are out-of-support: Windows 7, Windows 8.1, Windows 10 (all versions except 21H2 and 22H2).

Server operating systems

[Expand table](#)

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
Windows Server 2022	64-bit	.NET Framework 4.8	.NET Framework 4.8.1
Windows Server 2019 [†]	64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows Server, version 1809 [†]	64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows Server, version 1803 [†]	64-bit	.NET Framework 4.7.2	.NET Framework 4.8
Windows Server, version 1709 [†]	64-bit	.NET Framework 4.7.1	.NET Framework 4.7.2
Windows Server 2016 [†]	64-bit	.NET Framework 4.6.2	.NET Framework 4.7
			.NET Framework 4.7.1
			.NET Framework 4.7.2
			.NET Framework 4.8
Windows Server 2012 R2 [†]	64-bit	.NET Framework 4.5.1	.NET Framework 4.6.2
			.NET Framework 4.7

Operating system	Compatible editions	Preinstalled with the OS	Installable separately
			.NET Framework 4.7.1
			.NET Framework 4.7.2
			.NET Framework 4.8
Windows Server 2012 (64-bit edition)†	64-bit	.NET Framework 4.5	.NET Framework 4.6.2
			.NET Framework 4.7
			.NET Framework 4.7.1
			.NET Framework 4.7.2
			.NET Framework 4.8
Windows Server 2008 R2 SP1‡§	64-bit	--	.NET Framework 4.6.2
			.NET Framework 4.7
			.NET Framework 4.7.1
			.NET Framework 4.7.2
			.NET Framework 4.8
Windows Server 2008 SP2‡§	32-bit and 64-bit	--	.NET Framework 4.6.2

†This operating system is out-of-support. ‡On Windows Server 2008 R2, .NET Framework isn't supported for Itanium-based systems. §On Windows Server 2008 SP2, .NET Framework isn't supported in the Server Core role.

See also

- [Installation guide](#)
- [Getting started](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)

 **Collaborate with us on GitHub**

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Installation guide

Article • 05/16/2024

You can install .NET Framework on various Windows versions.

ⓘ Note

.NET Framework [is serviced monthly](#) with security and reliability bug fixes. .NET Framework will continue to be included with Windows, with no plans to remove it. You don't need to migrate your .NET Framework apps, but for new development, use [.NET 6 or later](#).

If you need to install .NET Framework 2.0 through 3.5, see [Install .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8](#).

Supported Windows versions

- [Windows 11 \(.NET Framework 4.8 included\)](#)
- [Windows Server 2022 \(.NET Framework 4.8 included\)](#)
- [Windows Server 2019 \(.NET Framework 4.7.2 included\)](#)
- [Windows 10 and Windows Server 2016](#)
- [Windows 8.1 and Windows Server 2012 R2](#)
- [Windows 8 and Windows Server 2012](#)

Unsupported Windows versions

- [Windows XP and Windows Server 2003](#)
- [Windows 7 and Windows Server 2008 R2](#)
- [Windows Vista and Windows Server 2008](#)

See also

- [Download .NET Framework](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install .NET Framework for developers](#)
- [Deploy .NET Framework for developers](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Install .NET Framework for developers

Article • 08/30/2022

.NET is an integral part of many apps running on Windows and provides common functionality for those apps to run. For developers, .NET Framework provides a comprehensive and consistent programming model for building apps that have visually stunning user experiences and seamless and secure communication.

ⓘ Note

This article is intended for **developers** who either want to install .NET Framework on their own system or who want to install it with their applications. For **users** interested in installing .NET Framework, see the individual articles that discuss installing .NET Framework on specific operating systems, such as [Install .NET Framework on Windows 10 and Windows Server 2016](#).

This article provides links for installing all versions of .NET Framework from .NET Framework 4.5 to .NET Framework 4.8.1 on your computer. If you're a developer, you can also use these links to download and redistribute .NET Framework with your apps. For information on deploying a version of .NET Framework with your app, see [.NET Framework deployment guide for developers](#).

ⓘ Important

.NET Framework content previously digitally signed using certificates that use the SHA1 algorithm, will be retired in order to support evolving industry standards.

The following versions of .NET Framework will reach end-of-support on *April 26, 2022*: 4.5.2, 4.6, and 4.6.1. After this date, security fixes, updates, and technical support for these versions will no longer be provided.

If you're using .NET Framework 4.5.2, 4.6, or 4.6.1, update your deployed runtime to a more recent version, such as [.NET Framework 4.6.2](#), before *April 26, 2022* in order to continue to receive updates and technical support.

Updated SHA2 signed installers will be available for .NET Framework 3.5 SP1, and 4.6.2 through 4.8. For more information, see the [SHA1 retirement plan](#), the [.NET 4.5.2, 4.6, and 4.6.1 lifecycle update blog post](#), and the [FAQ](#).

ⓘ Important

All .NET Framework versions since .NET Framework 4 are in-place updates, so only a single 4.x version can be present on a system. In addition, particular versions of .NET Framework are preinstalled on some versions of the Windows operating system. This means that:

- If there's a later 4.x version installed on the machine already, you can't install a previous 4.x version.
- If the OS comes preinstalled with a particular .NET Framework version, you can't install a previous 4.x version on the same machine.
- If you install a later version, you don't have to first uninstall the previous version.

For more information about versions of .NET Framework and how to determine which versions are installed on a computer, see [Versions and Dependencies](#) and [How to: Determine Which .NET Framework Versions Are Installed](#).

 **Note**

For information on .NET Framework 3.5, see [Install the .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8](#).

Use the following table for quick links, or read further for details. To view the system requirements for .NET Framework before installation, see [System Requirements](#). For help with troubleshooting, see [Troubleshooting](#).

.NET Framework version	Installer (Developer Pack and Runtime)	Platform support
4.8.1	.NET Framework 4.8.1	<p>Included in: Visual Studio 2022 (version 17.3)</p> <p>You can install on:</p> <p>Windows 11 Windows 10 version 21H2 Windows 10 version 21H1 Windows 10 version 20H2 Windows Server 2022</p> <p>(for a full list, see system requirements)</p>

.NET Framework version	Installer (Developer Pack and Runtime)	Platform support
4.8	.NET Framework 4.8	<p>Included in:</p> <ul style="list-style-type: none"> Windows 11 Windows 10 May 2019 Update (and later versions) Visual Studio 2019 (version 16.3) <p>You can install on:</p> <ul style="list-style-type: none"> Windows 10 October 2018 Update Windows 10 April 2018 Update Windows 10 Fall Creators Update Windows 10 Creators Update Windows 10 Anniversary Update Windows 8.1 and earlier Windows Server 2022 Windows Server 2019 Windows Server, Version 1809 Windows Server, Version 1803 <p>(for a full list, see system requirements)</p>
4.7.2	.NET Framework 4.7.2	<p>Included in:</p> <ul style="list-style-type: none"> Windows 10 October 2018 Update Windows 10 April 2018 Update Windows Server 2019 Windows Server, Version 1809 Windows Server, Version 1803 Visual Studio 2017 (15.8 update) <p>You can install on:</p> <ul style="list-style-type: none"> Windows 10 Fall Creators Update Windows 10 Creators Update Windows 10 Anniversary Update Windows 8.1 and earlier Windows Server, version 1709 and earlier <p>(for a full list, see system requirements)</p>

.NET Framework version	Installer (Developer Pack and Runtime)	Platform support
4.7.1	.NET Framework 4.7.1	<p>Included in:</p> <p>Windows 10 Fall Creators Update Windows Server, version 1709 Visual Studio 2017 (15.5 update)</p> <p>You can install on:</p> <p>Windows 10 Creators Update Windows 10 Anniversary Update Windows 8.1 and earlier Windows Server 2016 and earlier (for a full list, see system requirements)</p>
4.7	.NET Framework 4.7	<p>Included in:</p> <p>Windows 10 Creators Update Visual Studio 2017 (15.3 update)</p> <p>You can install on:</p> <p>Windows 10 Anniversary Update Windows 8.1 and earlier Windows Server 2016 and earlier (for a full list, see system requirements)</p>
4.6.2	.NET Framework 4.6.2	<p>Included in:</p> <p>Windows 10 Anniversary Update</p> <p>You can install on:</p> <p>Windows 10 November Update Windows 10 Windows 8.1 and earlier Windows Server 2012 R2 and earlier (for a full list, see system requirements)</p>

.NET Framework version	Installer (Developer Pack and Runtime)	Platform support
4.6.1	.NET Framework 4.6.1	<p>Included in:</p> <p>Visual Studio 2015 Update 2</p> <p>You can install on:</p> <p>Windows 10 Windows 8.1 and earlier Windows Server 2012 R2 and earlier (for a full list, see system requirements)</p>
4.6	.NET Framework 4.6	<p>Included in:</p> <p>Windows 10 Visual Studio 2015</p> <p>You can install on:</p> <p>Windows 8.1 and earlier Windows Server 2012 R2 and earlier (for a full list, see system requirements)</p>
4.5.2	.NET Framework 4.5.2	<p>You can install on:</p> <p>Windows 8.1 and earlier Windows Server 2012 R2 and earlier (for a full list, see system requirements)</p>
4.5.1	.NET Framework 4.5.1	<p>Included in:</p> <p>Windows 8.1 Windows Server 2012 R2 Visual Studio 2013</p> <p>You can install on:</p> <p>Windows 8 and earlier Windows Server 2012 and earlier (for a full list, see system requirements)</p>

.NET Framework version	Installer (Developer Pack and Runtime)	Platform support
4.5	.NET Framework 4.5	<p>Included in:</p> <p>Windows 8 Windows Server 2012 Visual Studio 2012</p> <p>You can install on:</p> <p>Windows 7 and earlier Windows Server 2008 SP2 and earlier (for a full list, see system requirements)</p>

Important

Starting with Visual Studio 2022, Visual Studio no longer includes .NET Framework components for .NET Framework 4.0 - 4.5.1 because these versions are no longer supported. Visual Studio 2022 and later versions can't build apps that target .NET Framework 4.0 through .NET Framework 4.5.1. To continue building these apps, you can use Visual Studio 2019 or an earlier version.

You can install the **Developer Pack** for a specific version of the .NET Framework, if one is available, on all supported platforms.

Developer Packs only target a specific version of .NET Framework and don't include previous versions. For example, the .NET Framework 4.8 Developer Pack doesn't include .NET Framework 4.7.

You can install the **Web or Offline** installer on:

- Windows 8.1 and earlier
- Windows Server 2012 R2 and earlier

For a full list, see [System Requirements](#).

For a general introduction to .NET Framework for both users and developers, see [Getting Started](#). For information about deploying .NET Framework with your app, see the [deployment guide](#). To read about the architecture and key features of .NET Framework, see the [overview](#).

Installation choices

Install a developer targeting pack to develop against the most recent version of .NET Framework in Visual Studio or another development environment, or download the .NET Framework redistributable for distribution with your app or control.

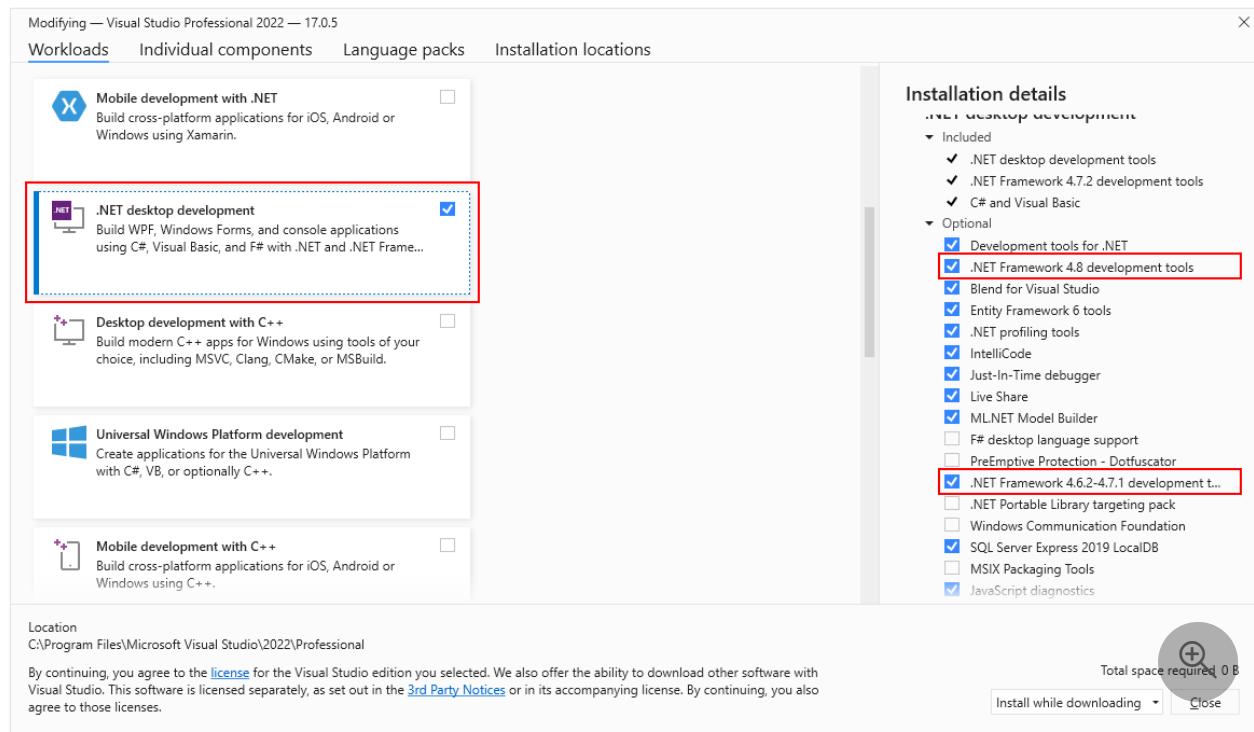
To install the .NET Framework Developer Pack or Targeting Pack

A *targeting pack* lets your app target a specific version of .NET Framework when developing in Visual Studio and some other development environments. A *developer pack* includes a specific version of .NET Framework and its accompanying SDK along with its corresponding targeting pack.

The developer pack for .NET Framework 4.5.1 or 4.5.2, the targeting pack for .NET Framework 4.6, and the developer pack for .NET Framework 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, or 4.8 provides a particular .NET Framework's version of the reference assemblies, language packs, and IntelliSense files for use in an integrated development environment such as Visual Studio. If you're using Visual Studio, the developer pack or targeting pack also adds the installed version of .NET Framework to the target choices when you create a new project. Choose one of the following:

- [.NET Framework 4.8.1 ↗](#)
- [.NET Framework 4.8 ↗](#)
- [.NET Framework 4.7.2 ↗](#)
- [.NET Framework 4.7.1 ↗](#)
- [.NET Framework 4.7 ↗](#)
- [.NET Framework 4.6.2 ↗](#)
- [.NET Framework 4.6.1 ↗](#)
- [.NET Framework 4.6 ↗](#)
- [.NET Framework 4.5.2 ↗](#) to install version 4.5.2 on Windows 8.1 or earlier, Visual Studio 2013, Visual Studio 2012, or other IDEs.
- [.NET Framework 4.5.1 ↗](#) to install version 4.5.1 on Visual Studio 2012 or other IDEs.

From the developer pack download page, choose **Download**. Next, choose **Run** or **Save**, and follow the instructions when prompted. You can also install the developer pack or targeting pack for a specific version of .NET Framework by selecting it from the optional components in the **.NET desktop development** workload in the Visual Studio Installer, as the following figure shows.



When you target a particular version of .NET Framework, your application is built by using the reference assemblies that are included with that version's developer pack. At run time, assemblies are resolved from the Global Assembly Cache, and the reference assemblies are not used.

When building an application from Visual Studio or using MSBuild from the command line, MSBuild may display error MSB3644, "The reference assemblies for framework "*framework-version*" were not found." To address the error, download the developer pack or the targeting pack for that version of .NET Framework.

To install or download the .NET Framework redistributable

Installers download .NET Framework components for an app or control that targets those versions of .NET Framework. These components must be installed on each computer where the app or control runs. These installers are redistributable, so you can include them in the setup program for your app.

The download page is provided in several languages, but most of the downloads are provided in English only. For additional language support, you must install a language pack.

Two types of redistributable installers are available:

- **Web installer** (web bootstrapper) downloads the required components and the language pack that matches the operating system of the installation computer from the web. This package is much smaller than the offline installer but requires a

consistent Internet connection. You can download the [standalone language packs](#) to install additional language support.

- **Offline installer** (standalone redistributable) contains all the required components for installing .NET Framework but doesn't contain language packs. This download is larger than the web installer. The offline installer doesn't require an internet connection. After you run the offline installer, you can download the [standalone language packs](#) to install language support. Use the offline installer if you can't rely on having a consistent Internet connection.

Both web and offline installers are designed for x86-based and x64-based computers (see [system requirements](#)), but do not support Itanium-based computers.

1. Open the download page for the .NET Framework version you want to install:

- [.NET Framework 4.8.1 ↗](#)
- [.NET Framework 4.8 ↗](#)
- [.NET Framework 4.7.2 ↗](#)
- [.NET Framework 4.7.1 ↗](#)
- [.NET Framework 4.7 ↗](#)
- [.NET Framework 4.6.2 ↗](#)
- [.NET Framework 4.6.1 ↗](#)
- [.NET Framework 4.6 ↗](#)
- [.NET Framework 4.5.2 ↗](#)
- [.NET Framework 4.5.1 ↗](#)
- [.NET Framework 4.5 ↗](#)

2. Select the language for the download page. This option does not download the localized resources of .NET Framework; it only affects the text displayed on the download page.

3. Choose **Download**.

4. If prompted, select the download that matches your system architecture, and then choose **Next**.

5. When the download prompt appears, do *one* of the following:

- If you want to install .NET Framework on your computer, choose **Run**, and then follow the prompts on your screen.
- If you want to download .NET Framework for redistribution, choose **Save**, and then follow the prompts on your screen.

6. If you want to download resources for additional languages, follow the instructions in the next section to install one or more language packs.

 **Note**

If you encounter any problems during the installation, see [Troubleshooting](#).

Installation notes:

- .NET Framework 4.5 and later versions replace .NET Framework 4.0. When you install these versions on a system that has .NET Framework 4 installed, the assemblies are replaced.
- Uninstalling .NET Framework 4.5 or later versions also removes pre-existing .NET Framework 4 files. If you want to go back to .NET Framework 4, you must reinstall it and any updates to it. See [Installing the .NET Framework 4](#).
- You must have administrative credentials to install .NET Framework 4.5 or later versions.
- The .NET Framework 4.5 redistributable was updated on October 9, 2012 to correct an issue related to an improper timestamp on a digital certificate, which caused the digital signature on files produced and signed by Microsoft to expire prematurely. If you previously installed the .NET Framework 4.5 redistributable package dated August 16, 2012, we recommend that you update your copy with the latest redistributable from the [.NET Framework download page](#). For more information about this issue, see [Microsoft Security Advisory 2749655](#).

To install language packs

Language packs are executable files that contain the localized resources (such as translated error messages and UI text) for supported languages. If you don't install a language pack, .NET Framework error messages and other text are displayed in English. Note that the web installer automatically installs the language pack that matches your operating system, but you can download additional language packs to your computer. The offline installers don't include any language packs.

 **Important**

The language packs don't contain the .NET Framework components that are required to run an app, so you must run the web or offline installer before you

install a language pack. If you have already installed a language pack, uninstall it, install the .NET Framework, and then reinstall the language pack.

1. Open the language pack download page for the .NET Framework version you've installed:

- [.NET Framework 4.8.1](#)
- [.NET Framework 4.8](#)
- [.NET Framework 4.7.2](#)
- [.NET Framework 4.7.1](#)
- [.NET Framework 4.7](#)
- [.NET Framework 4.6.2](#)
- [.NET Framework 4.6.1](#)
- [.NET Framework 4.6](#)
- [.NET Framework 4.5.2](#)
- [.NET Framework 4.5.1](#)
- [.NET Framework 4.5](#)

2. In the language list, choose the language you want to download, and wait a few seconds for the page to reload in that language.

3. Choose **Download**.

The following table lists the supported languages.

Language	Culture
Arabic	ar
Czech	cs
Danish	da
Dutch	nl
Finnish	fi
English (USA)	en-US
French	fr
German	de
Greek	el
Hebrew	he

Language	Culture
Hungarian	hu
Italian	it
Japanese	ja
Korean	ko
Norwegian	no
Polish	pl
Portuguese (Brazil)	pt-BR
Portuguese (Portugal)	pt-PT
Russian	ru
Simplified Chinese	zh-CHS
Spanish	es
Swedish	sv
Traditional Chinese	zh-CHT
Turkish	tr

Next steps

- If you're new to .NET Framework, see the [overview](#) for an introduction to key concepts and components.
- For new features and improvements in .NET Framework 4.5 and all later versions, see [What's New](#).
- For detailed information about deploying .NET Framework with your app, see [Deployment Guide for Developers](#).
- For changes that affect the deployment of .NET Framework with your app, see [Reducing System Restarts During .NET Framework 4.5 Installations](#).
- For information about migrating your app from .NET Framework 4 to .NET Framework 4.5 or later versions, see the [migration guide](#).
- See [.NET Framework Reference Source](#) to browse through .NET Framework source code online. The reference source is also available on [GitHub](#). You can

download the reference source [↗](#) for offline viewing and step through the sources (including patches and updates) during debugging. For more information, see the blog entry [A new look for .NET Reference Source ↗](#).

See also

- [Deployment Guide for Developers](#)
- [Deployment Guide for Administrators](#)
- [Install the .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8](#)
- [Troubleshoot Blocked .NET Framework Installations and Uninstallations](#)

Install .NET Framework on Windows 11

Article • 12/01/2022

.NET Framework 4.8 is included with Windows 11, and runs any .NET Framework 4.x app.

.NET Framework 3.5

Follow the instructions to install [.NET Framework 3.5 on Windows 11](#).

.NET Framework 3.5 supports apps built for .NET Framework 2.0 through 3.5.

See also

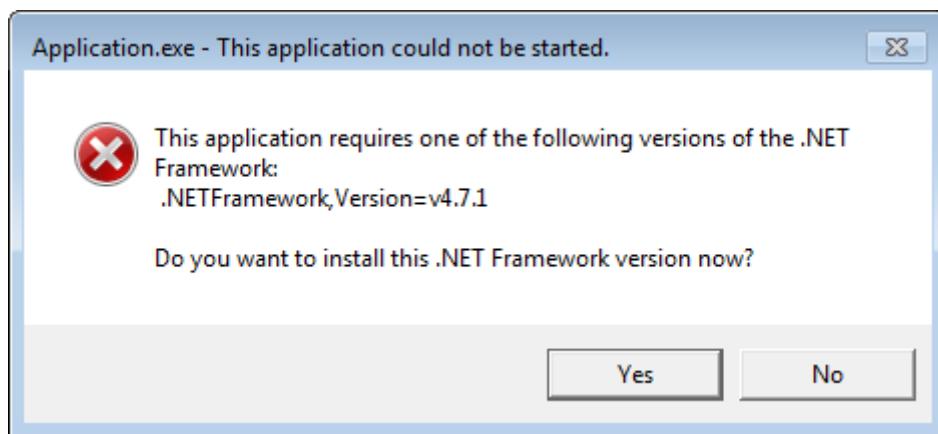
- [.NET Downloads](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install .NET Framework for developers](#)
- [Determine which .NET Framework versions are installed](#)

Install .NET Framework on Windows 10 and Windows Server 2016 and later

Article • 04/19/2023

The .NET Framework is required to run many applications on Windows. The instructions in this article should help you install the .NET Framework versions that you need. The [.NET Framework 4.8](#) is the latest available version.

You may have arrived on this page after trying to run an application and seeing a dialog on your machine similar to the following one:



.NET Framework 4.8

.NET Framework 4.8 is included with:

- [Windows 10 May 2021 Update](#)
- [Windows 10 October 2020 Update](#)
- [Windows 10 May 2020 Update](#)
- [Windows 10 November 2019 Update](#)
- [Windows 10 May 2019 Update](#)

[Download .NET Framework 4.8](#)

[.NET Framework 4.8](#) can be used to run applications built for the .NET Framework 4.0 through 4.8.

You can install [.NET Framework 4.8](#) on:

- Windows 10 October 2018 Update (version 1809)
- Windows 10 April 2018 Update (version 1803)
- Windows 10 Fall Creators Update (version 1709)

- Windows 10 Creators Update (version 1703)
- Windows 10 Anniversary Update (version 1607)
- Windows Server 2019
- Windows Server, version 1809
- Windows Server, version 1803
- Windows Server 2016

The .NET Framework 4.8 is not supported on:

- Windows 10 1507
- Windows 10 1511

If you're using Windows 10 1507 or 1511 and you want to install .NET Framework 4.8, you first need to upgrade to a later Windows 10 version.

.NET Framework 4.6.2

The [.NET Framework 4.6.2](#) is the latest supported .NET Framework version on Windows 10 1507 and 1511.

The .NET Framework 4.6.2 supports apps built for the .NET Framework 4.0 through 4.6.2.

.NET Framework 3.5

Follow the instructions to install [.NET Framework 3.5 on Windows 10](#).

The .NET Framework 3.5 supports apps built for the .NET Framework 1.0 through 3.5.

Additional information

.NET Framework 4.x versions are in-place updates to earlier versions. That means the following:

- You can only have one version of the .NET Framework 4.x installed on your machine.
- You cannot install an earlier version of the .NET Framework on your machine if a later version is already installed.
- 4.x versions of the .NET Framework can be used to run applications built for the .NET Framework 4.0 through that version. For example, .NET Framework 4.7 can be used to run applications built for the .NET Framework 4.0 through 4.7. The latest

version (the .NET Framework 4.8) can be used to run applications built with all versions of the .NET Framework starting with 4.0.

For a list of all the versions of the .NET Framework available to download, see the [.NET Downloads](#) page.

Help

If you cannot get the correct version of the .NET Framework installed, you can [contact Microsoft for help](#).

See also

- [.NET Downloads](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install the .NET Framework for developers](#)
- [Determine which .NET Framework versions are installed](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

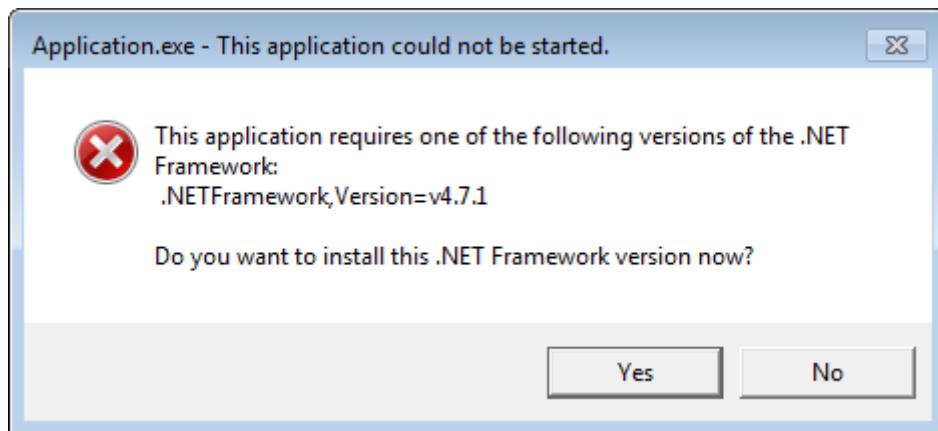
 [Open a documentation issue](#)

 [Provide product feedback](#)

Install .NET Framework on Windows 8.1 and Windows Server 2012 R2

Article • 10/12/2021

The .NET Framework is required to run many applications on Windows. You can use the following instructions to install it. You may have arrived on this page after trying to run an application and seeing the following dialog on your machine.



These instructions will help you install the .NET Framework versions you need. [.NET Framework 4.8](#) is the latest version. It is supported on Windows 8.1 and Windows Server 2012 R2. It's included with Windows 11 and in Windows 10 starting with the [May 2019 Update](#).

.NET Framework 4.8

[Download .NET Framework 4.8](#)

[.NET Framework 4.8](#) can be used to run applications built for .NET Framework 4.0 or later.

.NET Framework 3.5

Follow the instructions to install the [.NET Framework 3.5](#) on [Windows 8.1](#), [Windows 10](#), and [Windows 11](#).

The .NET Framework 3.5 supports apps built for .NET Framework 1.0 through 3.5.

Help

You can [contact Microsoft for help](#) if you cannot get the correct version of the .NET Framework installed.

See also

- [Download the .NET Framework ↗](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install the .NET Framework for developers](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

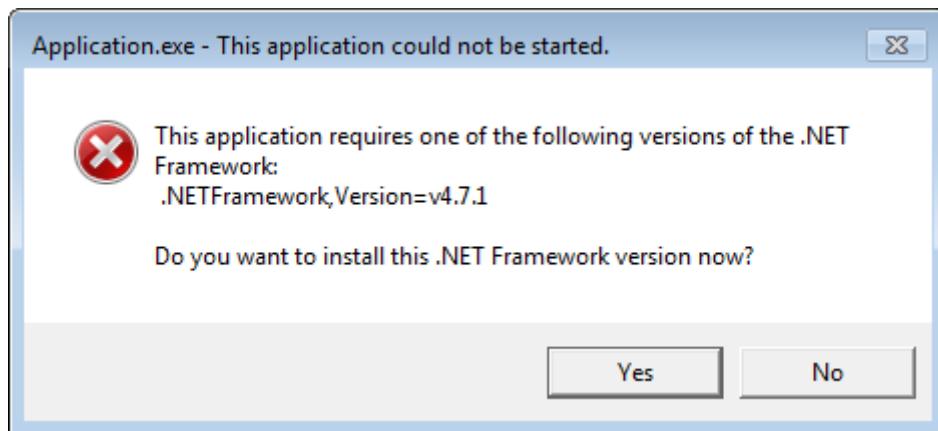
.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

Install .NET Framework on Windows 8 and Windows Server 2012

Article • 10/12/2021

The .NET Framework is required to run many applications on Windows. You can use the following instructions to install it. You may have arrived on this page after trying to run an application and seeing the following dialog on your machine.



These instructions will help you install the .NET Framework versions you need. [.NET Framework 4.8](#) is the latest version. It is supported on Windows Server 2012 but is not supported on Windows 8. It's included with Windows 11 and in Windows 10 starting with the [May 2019 Update](#).

.NET Framework 4.8

[Download .NET Framework 4.8](#)

.NET Framework 4.8 is supported on Windows Server 2012. It is not supported on Windows 8.

[.NET Framework 4.8](#) can be used to run applications built for .NET Framework 4.0 or later.

.NET Framework 4.6

The [.NET Framework 4.6](#) is the latest supported .NET Framework version on Windows 8.

The .NET Framework 4.6 supports apps built for .NET Framework 4.0 through 4.6.

.NET Framework 3.5

Follow the instructions to install the .NET Framework 3.5 on Windows 8, Windows 10, and Windows 11.

The .NET Framework 3.5 supports apps built for .NET Framework 1.0 through 3.5.

Help

You can [contact Microsoft for help](#) if you cannot get the correct version of the .NET Framework installed.

See also

- [Download the .NET Framework ↗](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install the .NET Framework for developers](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Install .NET Framework on Windows Server 2022

Article • 06/04/2024

.NET Framework 4.8 is included with Server 2022, and runs any .NET Framework 4.x app.

.NET Framework 4.8.1

.NET 4.8.1 is supported on Windows Server 2022.

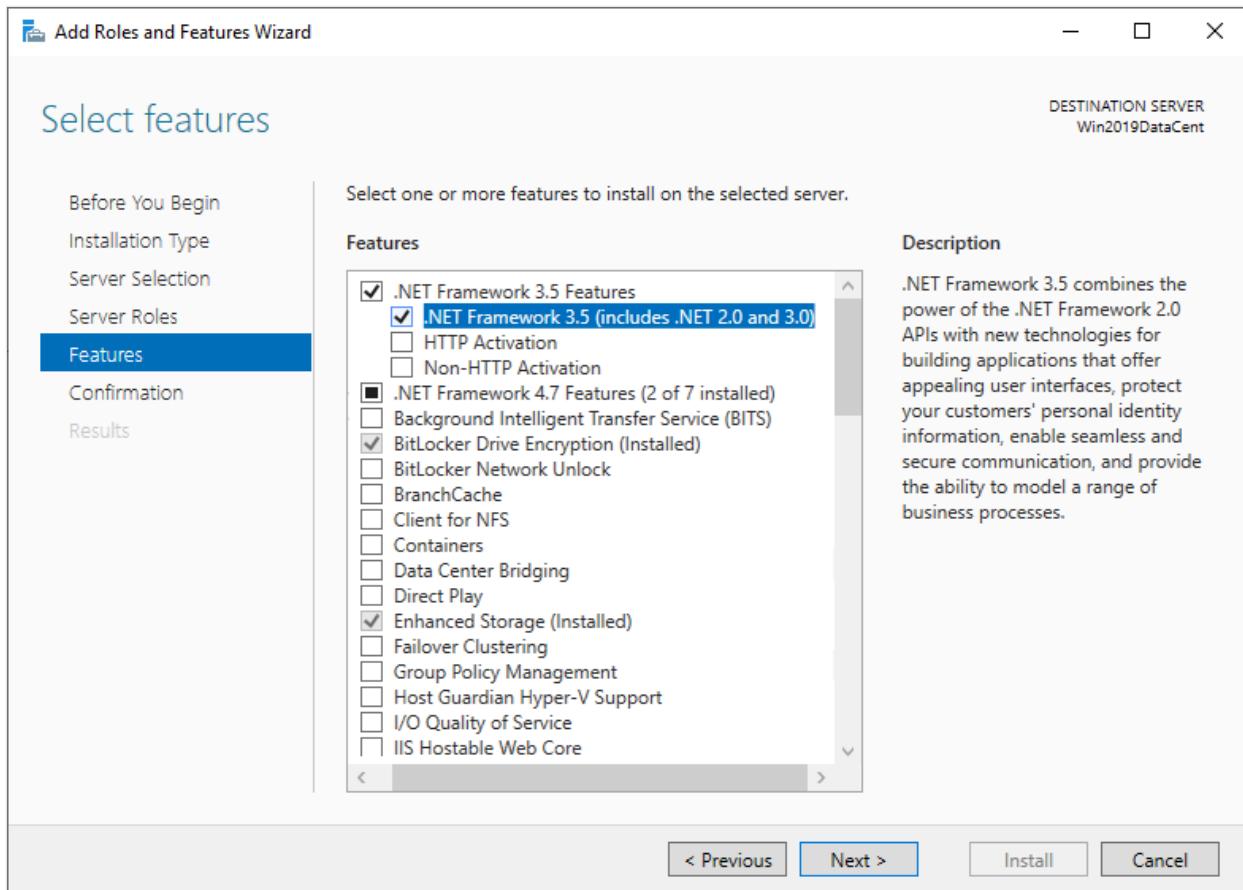
[Download .NET Framework 4.8.1](#)

.NET Framework 3.5

.NET Framework 3.5 supports apps built for .NET Framework 1.0 through 3.5.

Enable .NET Framework 3.5 through the **Add Roles and Features Wizard**.

1. Open the Start Menu.
2. Search for **Add Roles and Features Wizard** and open it.
3. Navigate through the wizard until you reach **Features**.
4. Select **.NET Framework 3.5 Features** in the list.
5. Finally, select **Install** to start installing .NET Framework 3.5.



See also

- [.NET Downloads ↗](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install .NET Framework for developers](#)
- [Determine which .NET Framework versions are installed](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Install .NET Framework on Windows Server 2019

Article • 06/28/2024

.NET Framework 4.7.2 is included with Server 2019, and runs any .NET Framework 4.x app.

.NET Framework 4.8

.NET Framework 4.8 is the last supported version of .NET Framework for Windows Server 2019.

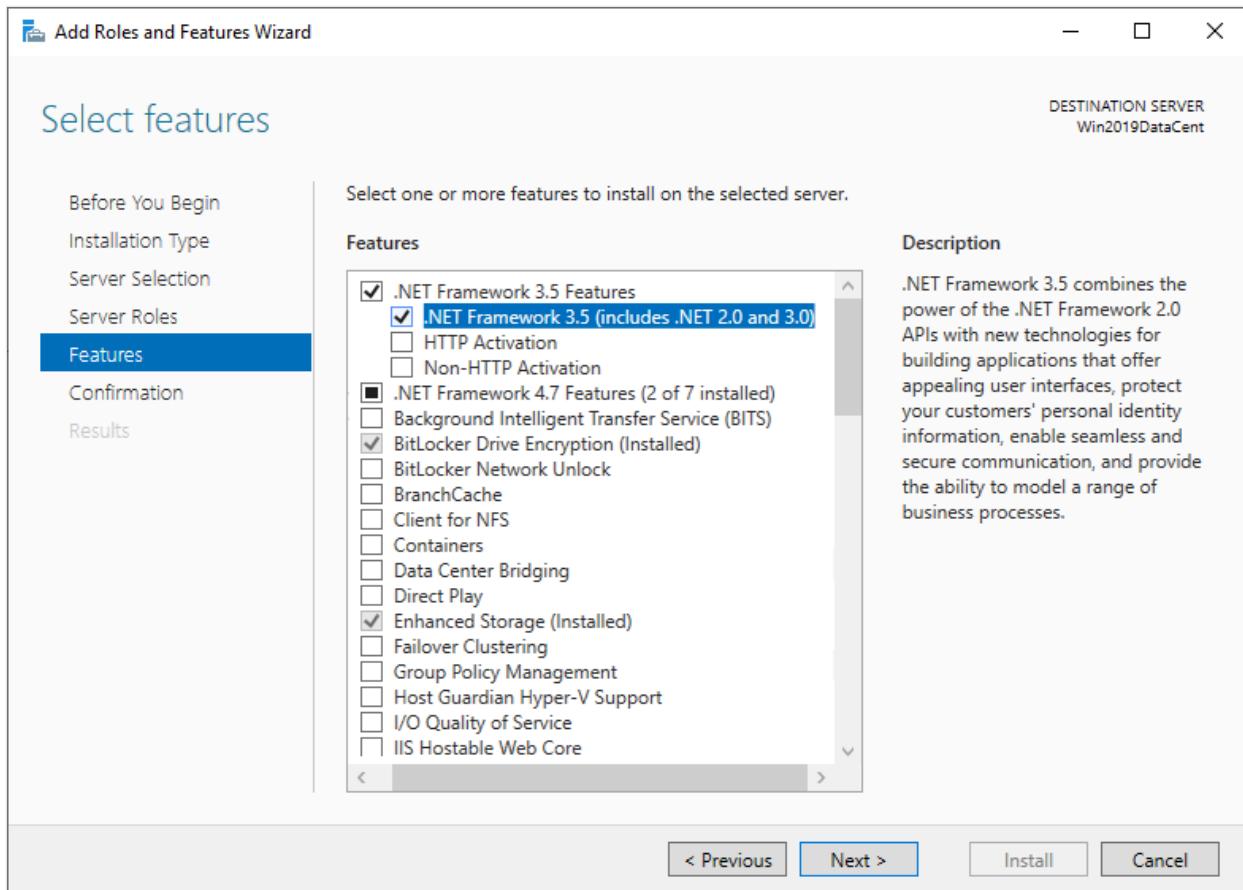
[Download .NET Framework 4.8.1](#)

.NET Framework 3.5

.NET Framework 3.5 supports apps built for .NET Framework 1.0 through 3.5.

Enable .NET Framework 3.5 through the **Add Roles and Features Wizard**.

1. Open the Start Menu.
2. Search for **Add Roles and Features Wizard** and open it.
3. Navigate through the wizard until you reach **Features**.
4. Select **.NET Framework 3.5 Features** in the list.
5. Finally, select **Install** to start installing .NET Framework 3.5.



See also

- [.NET Downloads ↗](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- [Install .NET Framework for developers](#)
- [Determine which .NET Framework versions are installed](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Repair .NET Framework

Article • 09/15/2021

In some situations, your .NET Framework installation can become damaged and require repairs. This might be the case if your app crashes right after you try to start it or if you cannot install newer .NET Framework versions.

You can repair your .NET Framework install using the [.NET Framework Repair Tool](#).

If your app still isn't starting after repairing .NET Framework, then the app might have a problem. In that case, you should contact the app publisher.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Troubleshoot blocked .NET Framework installations and uninstallations

Article • 08/30/2022

When you run the [web or offline installer](#) for the .NET Framework 4.5 or later versions, you might encounter an issue that prevents or blocks the installation of the .NET Framework. The following table lists possible blocking issues and provides links to troubleshooting information.

In Windows 8 and above, the .NET Framework is an operating system component and cannot be independently uninstalled. Updates to the .NET Framework appear in the **Installed Updates** tab of the Control Panel **Programs and Features** app. For operating systems on which the .NET Framework is not preinstalled, the .NET Framework appears in the **Uninstall or change a program** tab (or the **Add/Remove programs** tab) of the **Program and Features** app in Control Panel. For information on the Windows versions on which the .NET Framework is preinstalled, see [System Requirements](#).

Important

Because the 4.x versions of the .NET Framework are in-place updates, you cannot install an earlier version of the .NET Framework 4.x on a system that already has a later version installed. For example, on a system with Windows 10 Fall Creators Update, you cannot install the .NET Framework 4.6.2, since the .NET Framework 4.7.1 is preinstalled with the operating system.

You can determine which versions of the .NET Framework are installed on a system. See [How to: Determine Which .NET Framework Versions Are Installed](#) for more information.

In this table, 4.5.x refers to the .NET Framework 4.5 and its point releases, 4.5.1, and 4.5.2, 4.6.x refers to the .NET Framework 4.6 and its point releases, 4.6.1 and 4.6.2, 4.7.x refers to the .NET Framework 4.7 and its point releases, 4.7.1 and 4.7.2, and 4.8.x refers to .NET Framework 4.8 and 4.8.1.

Blocking message	For more information or to resolve the issue
Uninstalling the Microsoft .NET Framework may cause some applications to cease to function.	In general, you should not uninstall any versions of the .NET Framework that are installed on your computer, because an application you use may depend on a specific version of the .NET Framework. For more information, see The .NET Framework for users in the Getting Started guide.

Blocking message	For more information or to resolve the issue
<p>.NET Framework 4.5.x/4.6.x/4.7.x (ENU) or a later version is already installed on this computer.</p>	<p>No action necessary.</p> <p>To determine which versions of the .NET Framework are installed on a system, see How to: Determine Which .NET Framework Versions Are Installed.</p>
<p>The .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x (<i>language</i>) requires the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x. Please install the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x from the Download Center and rerun Setup.</p>	<p>You must install the English version of the specified .NET Framework release before installing a language pack. For more information, see the section on To install language packs in the installation guide.</p>
<p>Cannot install the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x. Other applications on your computer are not compatible with this program.</p>	<p>The most likely cause of this message is that a preview or RC version of the .NET Framework was installed. Uninstall the preview or RC version and rerun Setup.</p>
<p>-or-</p>	
<p>Other applications on your computer are not compatible with this program.</p>	
<p>.NET Framework 4.5.x/4.6.x/4.7.x/4.8.x cannot be uninstalled using this package. To uninstall .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x from your computer, go to Control Panel, choose Programs and Features, choose View installed updates, select Update for Microsoft Windows (KB2828152) and then choose Uninstall.</p>	<p>The package you are installing doesn't uninstall preview or RC releases of the .NET Framework.</p> <p>Uninstall the preview or RC release from Control Panel.</p>
<p>Cannot uninstall the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x. Other applications on your computer are dependent on this program.</p>	<p>In general, you shouldn't uninstall any versions of the .NET Framework from your computer, because an application you use may depend on a specific version of the .NET Framework. For more information, see The .NET Framework for users in the <i>Getting Started</i> guide.</p>

Blocking message	For more information or to resolve the issue
<p>The .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x redistributable does not apply to this operating system. Please download the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x for your operating system from the .NET Framework download page.</p>	<p>You may be trying to install .NET Framework 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, or 4.8.x on a platform that isn't supported, or you have chosen the installation package that does not include the components for all supported operating systems. Run the installation again by using the offline installer (for 4.5.1, for 4.5.2, for 4.6, for 4.6.1, for 4.6.2, for 4.7, for 4.7.1, for 4.7.2, for 4.8, or for 4.8.1). For more information, see the installation guide and system requirements for supported operating systems.</p>
<p>The update corresponding to KB<number> needs to be installed before you can install this product.</p>	<p>Installation of the .NET Framework requires that a KB update be installed before installing the .NET Framework. Install the update, and then begin the .NET Framework installation again.</p> <p>For example, installation of updated versions of the .NET Framework on Windows 8.1, Windows RT 8.1, and Windows Server 2012 R2 requires that the update corresponding to KB 2919355 be installed.</p>
<p>Your computer is currently running a Server Core installation of the Windows Server 2008 operating system. The .NET Framework 4.5.x requires a later release of the operating system. Please install Windows Server 2008 R2 SP1 or higher and rerun .NET Framework 4.5.x setup.</p>	<p>The .NET Framework 4.5.1 and 4.5.2 are supported in the Server Core role with Windows Server 2008 R2 SP1 or later. See System Requirements.</p>
<p>You do not have sufficient privileges to complete this operation for all users of this computer. Log on as an administrator and rerun Setup.</p>	<p>You must be an administrator on the computer to install the .NET Framework.</p>
<p>Setup cannot continue because a previous installation requires your computer to be restarted. Please restart your computer and rerun Setup.</p>	<p>A restart is sometimes required to fully complete an installation. Follow the instructions to restart your computer and rerun Setup.</p> <p>In rare cases, you may be asked to restart your system more than once if Windows has detected a number of missing updates and is restarting to install the next update in the queue.</p>
<p>.NET Framework Setup cannot be run in Program Compatibility Mode.</p>	<p>See the Program Compatibility Issues section later in this article.</p>

Blocking message	For more information or to resolve the issue
<p>.NET Framework 4.5.x/4.6.x/4.7.x/4.8.x has not been installed because the component store has been corrupted.</p>	<p>See Fix Windows Update errors by using the DISM or System Update Readiness tool for more information.</p>
<p>Setup cannot run because the Windows Installer Service is not available on this computer.</p>	<p>See "The Windows Installer Service Could Not Be Accessed" error when you try to install a program in Windows 7 or Windows Vista on the Microsoft Support website.</p>
<p>Setup may not run properly because the Windows Update Service is not available on this computer.</p>	<p>The computer may be configured to use Windows Server Update Services (WSUS) instead of Microsoft Windows Update. For more information, see the section for error code 0x800F0906 in .NET Framework 3.5 installation error: 0x800F0906, 0x800F081F, 0x800F0907.</p>
	<p>Also see How to update the Windows Update Agent to the latest version on the Microsoft Support website.</p>
<p>Setup may not run properly because the Background Intelligent Transfer Service (BITS) is not available on this computer.</p>	<p>See An update is available to fix a Background Intelligent Transfer Service (BITS) crash on a Windows Vista-based computer on the Microsoft Support website.</p>
<p>Setup may not run properly because Windows update encountered an error and displayed error code 0x80070643 or 0x643.</p>	<p>See .NET Framework update installation error: "0x80070643" or "0x643" on the Microsoft Support website.</p>
<p>The .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x is already a part of this operating system. You do not need to install the .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x redistributable.</p>	<p>No action.</p> <p>To determine which versions of the .NET Framework are installed on a system, see How to: Determine Which .NET Framework Versions Are Installed. See System Requirements for supported operating systems.</p>
<p>The .NET Framework 4.5.x/4.6.x/4.7.x/4.8.x is not supported on this operating system.</p>	<p>See System Requirements for supported operating systems.</p>
	<p>For failed installations of the .NET Framework on Windows 7, this message typically indicates that Windows 7 SP1 is not installed. On Windows 7 systems, the .NET Framework requires Windows 7 SP1. If you are on Windows 7 and have not yet installed Service Pack 1, you will need to do so before installing the .NET Framework. For information on installing Windows 7 SP1, see Learn how to install Windows 7 Service Pack 1 (SP1).</p>

Blocking message	For more information or to resolve the issue
<p>Your computer is currently running a Server Core installation of Windows Server 2008 operating system. The .NET Framework 4.5.x requires a full release of the operating system or Server Core 2008 R2 SP1. Please install the full version of Windows Server 2008 SP2 or Windows Server 2008 R2 SP1 or Server Core 2008 R2 SP1 and rerun .NET Framework 4.5.x Setup.</p>	<p>The .NET Framework is supported in the Server Core role with Windows Server 2008 R2 SP1 or later. See System Requirements.</p>
<p>The .NET Framework 4.5.x is already a part of this operating system but is currently turned off (Windows Server 2012 only).</p>	<p>Use Turn Windows features on or off in the Control Panel to turn on .NET Framework 4.5.x.</p>
<p>This setup program requires an x86 computer. It cannot be installed on x64 or IA64 computers.</p>	<p>See System Requirements.</p>
<p>This setup program requires x64 or x86 computer. It cannot be installed on IA64 computers.</p>	<p>See System Requirements.</p>

Program compatibility issues

The installation of the .NET Framework 4.5 or its point releases fails with a 1603 error code or blocks when it's running in Windows Program Compatibility mode. The **Program Compatibility Assistant** indicates that the .NET Framework might not have been installed correctly and prompts you to reinstall it by using the recommended setting (Program Compatibility mode). Program Compatibility mode could also have been set by the Program Compatibility Assistant on earlier failed or canceled attempts to run the .NET Framework Setup.

The .NET Framework installer cannot run in Program Compatibility mode. To resolve this blocking issue, you must use Registry Editor to ensure that the compatibility mode setting is not enabled system-wide:

1. Choose the **Start** button, and then choose **Run**.
2. In the **Run** dialog box, type "regedit", and then choose **OK**.
3. In Registry Editor, browse to the following subkeys:

- HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Persisted
 - HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Layers
4. In the Name column, look for .NET Framework 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, or 4.7.2 download names, depending on which version you are installing, and delete these entries. For download names, see [Install the .NET Framework for developers](#) article.
5. Rerun the .NET Framework installer for version 4.5, 4.5.1, 4.5.2, or 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, or 4.7.2.

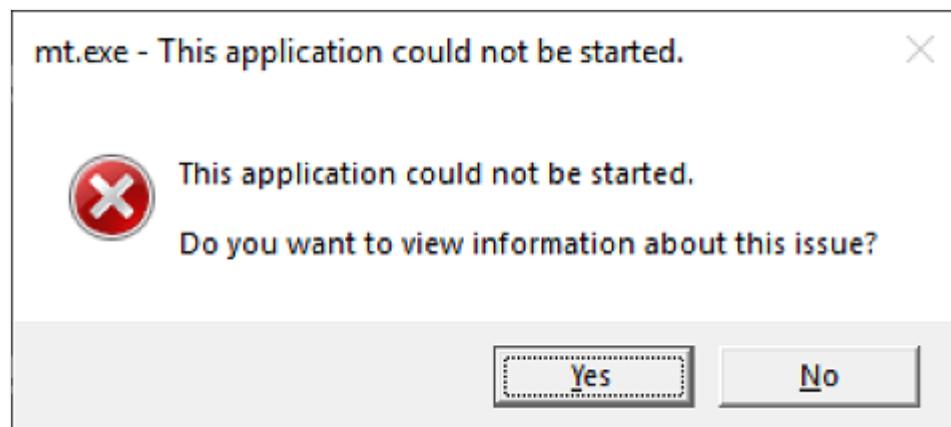
See also

- [Install the .NET Framework for developers](#)
- [How to: Determine Which .NET Framework Versions Are Installed](#)
- [Versions and Dependencies](#)

"This application could not be started" error when running a .NET Framework application

Article • 02/16/2023

When you attempt to run a .NET Framework application, you may receive the "This application could not be started" error message. When this error is caused by an installed version of .NET Framework not being detected, or by .NET Framework being corrupted, use this article to try to solve that problem.

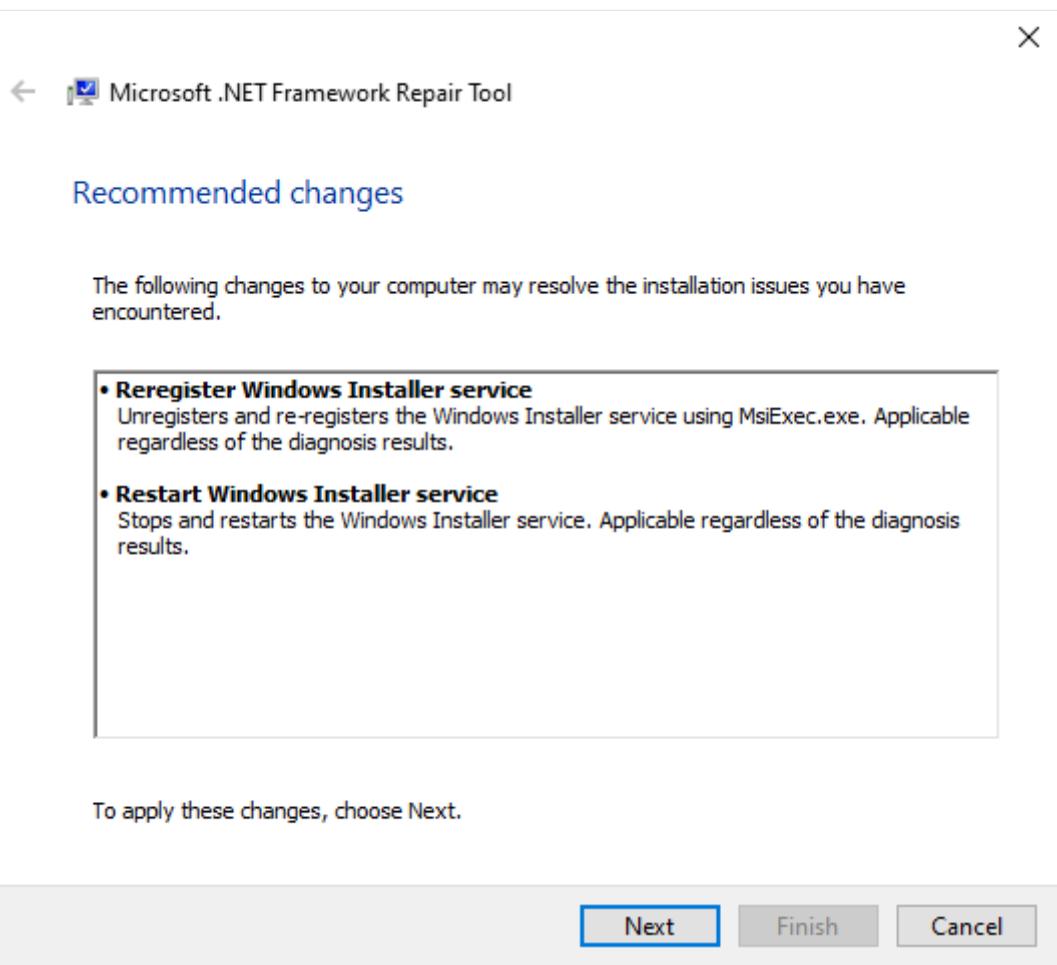


If you still can't run the application after completing all the steps in this article, then the issue may be caused by some other reason, like a corrupted file system, missing dependencies, or a problem with the application. In that case, you can try contacting the app publisher or post a question to [Microsoft Support Community](#) or [Microsoft Q&A](#) for more help.

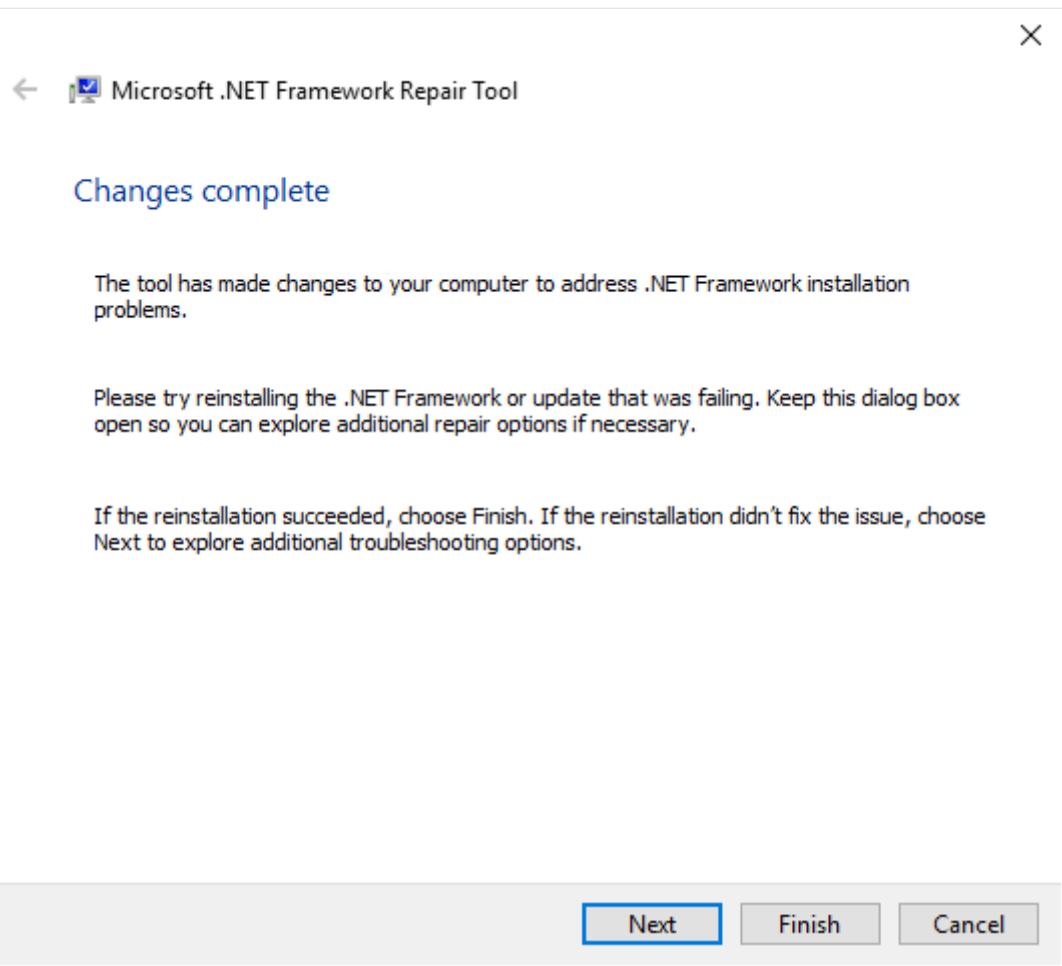
How to fix the error

To address this issue so that you can run your application, do the following:

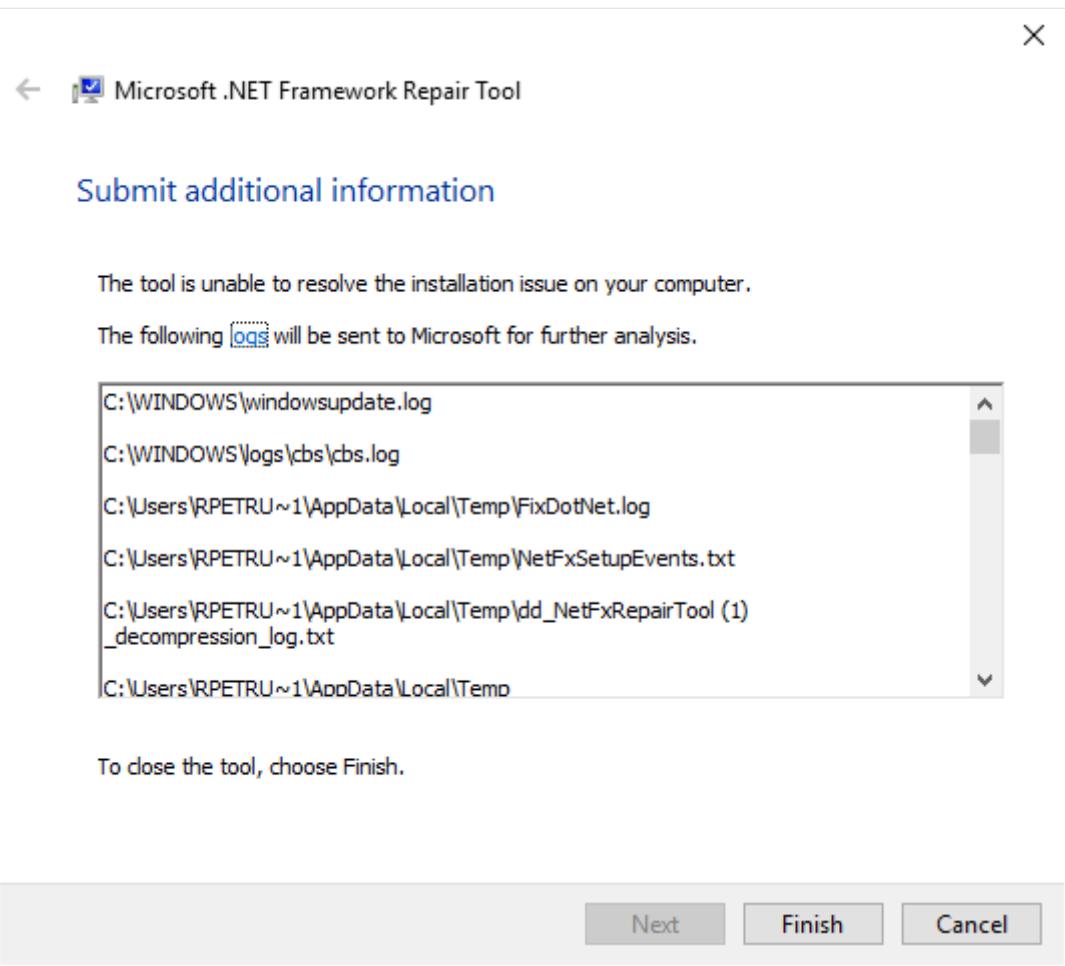
1. Download the [.NET Framework Repair Tool \(NetFxRepairTool.exe\)](#). The tool runs automatically when the download completes.
2. If the .NET Framework Repair Tool recommends any additional action, such as those shown in the following figure, select **Next**.



3. The .NET Framework Repair Tools displays a dialog box shown in the following figure to indicate that changes are complete. Leave the dialog box open while you to try rerun your application. This should succeed if the .NET Framework Repair Tool has identified and corrected a corrupted .NET Framework installation.



4. If your application runs successfully, select the **Finish** button. Otherwise, select the **Next** button.
5. If you selected the **Next** button, the .NET Framework Repair Tool displays a dialog box like the following. Select the **Finish** button to send diagnostic information to Microsoft.



6. If you still cannot run the application, install the latest version of .NET Framework that's supported by your version of Windows, as shown in the following table.

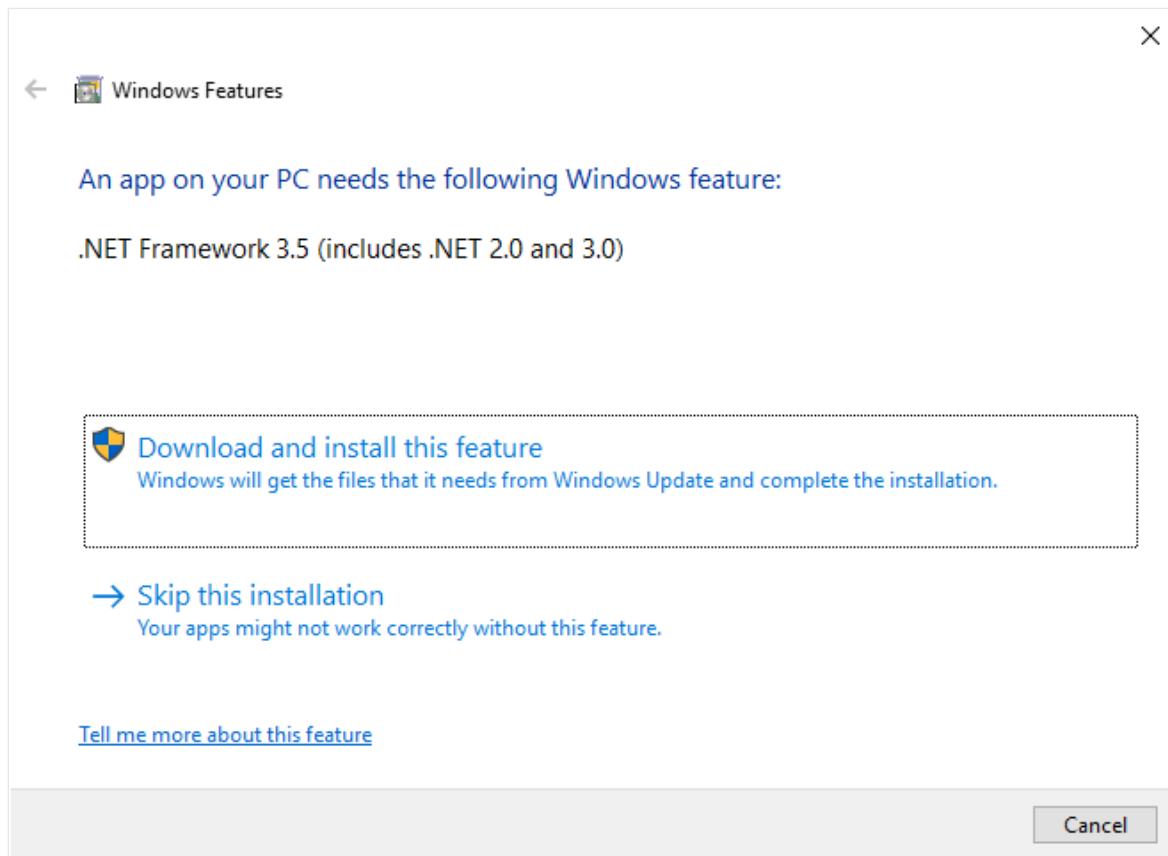
Windows version	.NET Framework installation
Windows 10 Anniversary Update and later versions	.NET Framework 4.8 Runtime
Windows 10, Windows 10 November Update	.NET Framework 4.6.2
Windows 8.1	.NET Framework 4.8 Runtime
Windows 8	.NET Framework 4.6.1
Windows 7 SP1	.NET Framework 4.8 Runtime
Windows Vista SP2	.NET Framework 4.6

Note

.NET Framework 4.8 is preinstalled on Windows 11 and Windows 10 May 2019 Update and later versions.

7. Attempt to launch the application.

8. In some cases, you may see a dialog box like the following, which asks you to install .NET Framework 3.5. Select **Download and install this feature** to install .NET Framework 3.5, then launch the application again.



See also

- [.NET Framework System Requirements](#)
- [.NET Framework installation guide](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)

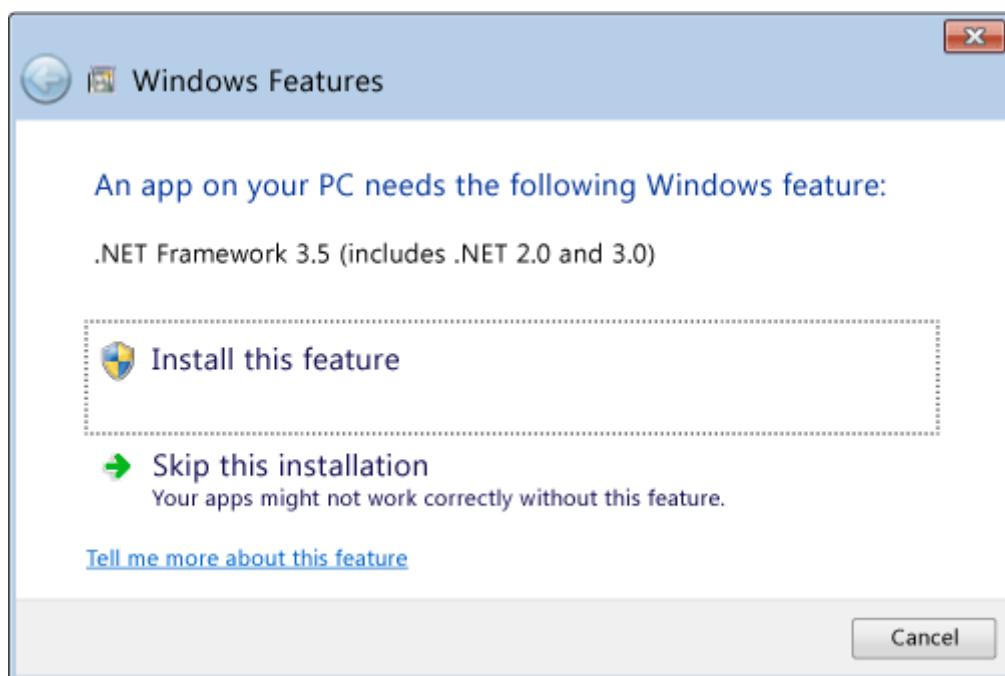
Install .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8

Article • 06/04/2024

You may need the .NET Framework 3.5 to run an app on Windows 11, Windows 10, Windows 8.1, and Windows 8. You can also use these instructions for earlier Windows versions.

Install .NET Framework 3.5 on Demand

You may see the following configuration dialog if you try to run an app that requires .NET Framework 3.5. Choose **Install this feature** to enable .NET Framework 3.5. This option requires an Internet connection.



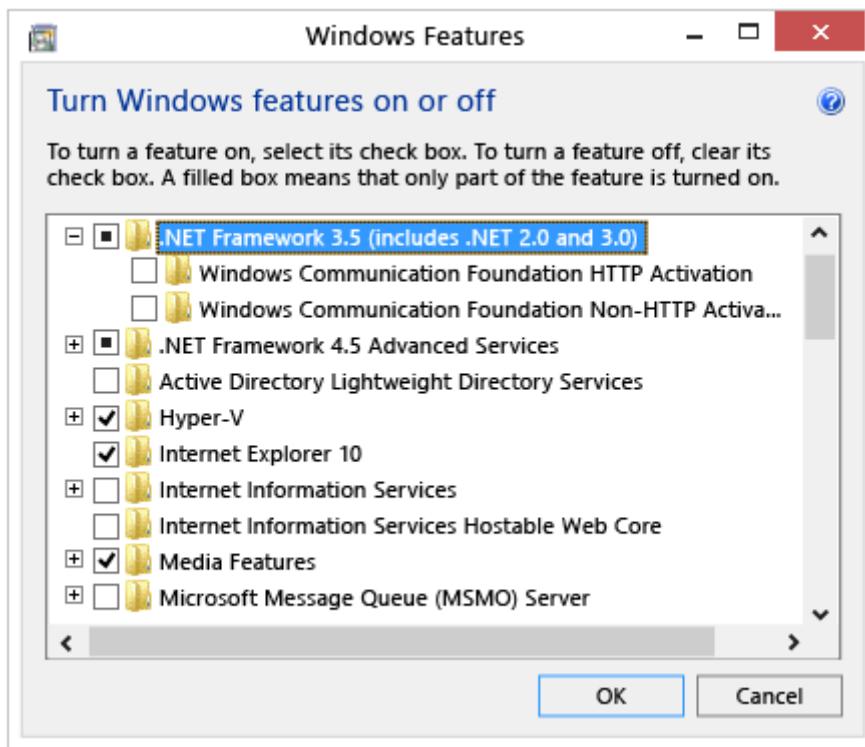
Why am I getting this pop-up?

The .NET Framework is created by Microsoft and provides an environment for running applications. There are different versions available. Many companies develop their apps to run using the .NET Framework, and these apps target a specific version. If you see this pop-up, you're trying to run an application that requires .NET Framework version 3.5, but that version is not installed on your system.

Enable .NET Framework 3.5 in Control Panel

You can enable the .NET Framework 3.5 through the Windows Control Panel. This option requires an Internet connection.

1. Press the Windows key  on your keyboard, type "Windows Features", and press Enter. The **Turn Windows features on or off** dialog box appears.
2. Select the **.NET Framework 3.5 (includes .NET 2.0 and 3.0)** check box, select OK, and reboot your computer if prompted.



You don't need to select the child items for **Windows Communication Foundation (WCF) HTTP Activation** and **Windows Communication Foundation (WCF) Non-HTTP Activation** unless you're a developer or server administrator who requires this functionality.

Download the offline installer

For Windows versions prior to Windows 10, the .NET Framework 3.5 SP1 offline installer is available on the [.NET Framework 3.5 SP1 Download page](#).

Troubleshoot the installation

During installation, you may encounter error 0x800f0906, 0x800f0907, 0x800f081f, or 0x800F0922, in which case refer to [.NET Framework 3.5 installation error: 0x800f0906](#),

0x800f0907, or 0x800f081f [🔗](#) to see how to resolve these issues.

If you still can't resolve your installation issue or you don't have an Internet connection, you can try installing it using your Windows installation media. For more information, see [Deploy .NET Framework 3.5 by using Deployment Image Servicing and Management \(DISM\)](#). If you're using Windows 7, Windows 8.1, the latest release Windows 10, or Windows 11, but you don't have the installation media, create an up-to-date installation media here: [Create installation media for Windows](#) [🔗](#). Additional information about Windows 11 and Windows 10 Features on Demand: [Features on Demand](#).

Warning

If you're not relying on Windows Update as the source for installing .NET Framework 3.5, you must ensure to strictly use sources from the same corresponding Windows operating system version. Using sources from a different Windows operating system version will either install a mismatched version of .NET Framework 3.5 or cause the installation to fail, leaving the system in an unsupported and unserviceable state.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Run .NET Framework 1.1 apps on Windows 8, Windows 8.1, Windows 10, or Windows 11

Article • 10/12/2021

.NET Framework 1.1 is not supported on the Windows 8, Windows 8.1, Windows Server 2012, Windows Server 2012 R2, Windows 10, or Windows 11 operating systems. In some cases, .NET Framework 1.1 is required for an app to run. In those cases, contact your independent software vendor (ISV) to have the app upgraded to run on .NET Framework 3.5 SP1 or a later version. For more information, see [Migrating from .NET Framework 1.1](#).

Install .NET Framework 1.1 from a CD or download center

It isn't possible to manually install .NET Framework 1.1 on Windows 8, Windows 8.1, Windows Server 2012, Windows Server 2012 R2, Windows 10, or Windows 11 from a CD or download center. It's no longer supported. If you try to install the package, the following error message is displayed: "Setup cannot continue because this version of the .NET Framework is incompatible with a previously installed one." To solve this problem, install [.NET Framework 3.5 SP1](#). This version includes .NET Framework 2.0 (the release that follows .NET Framework 1.1), which is supported on Windows 8, Windows 8.1, Windows 10, and Windows 11. You should always try to install the app first to determine if it will automatically be updated to a later version of .NET Framework. If it doesn't, contact your ISV for an app update.

See also

- [Migrating from the .NET Framework 1.1](#)
- [Install the .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8](#)

Migrate to .NET Framework 4.8, 4.7, and 4.6.2

Article • 01/02/2023

If you created your app using an earlier version of .NET Framework, you can generally upgrade it to .NET Framework 4.6.2, .NET Framework 4.7 and its point releases (4.7.1 and 4.7.2), or .NET Framework 4.8 easily. Open your project in Visual Studio. If your project was created in an earlier version of Visual Studio, the **Project Compatibility** dialog box automatically opens. For more information about upgrading a project in Visual Studio, see [Port, Migrate, and Upgrade Visual Studio Projects](#) and [Visual Studio 2022 Platform Targeting and Compatibility](#).

However, some changes in .NET Framework require changes to your code. You may also want to take advantage of functionality that is new in .NET Framework 4.6.2, in .NET Framework 4.7 and its point releases, or in .NET Framework 4.8. Making these types of changes to your app for a new version of .NET Framework is typically referred to as *migration*. If your app doesn't have to be migrated, you can run it in .NET Framework 4.6.2 or a later version without recompiling it.

Migration resources

Review the following documents before you migrate your app from earlier versions of .NET Framework to version 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, or 4.8.1:

- See [Versions and Dependencies](#) to understand the CLR version underlying each version of the .NET Framework and to review guidelines for targeting your apps successfully.
- Review [Application compatibility](#) to find out about runtime and retargeting changes that might affect your app and how to handle them.
- Review [What's Obsolete in the Class Library](#) to determine any types or members in your code that have been made obsolete, and the recommended alternatives.
- See [What's New](#) for descriptions of new features that you may want to add to your app.

See also

- [Application compatibility](#)

- Migrating from the .NET Framework 1.1
- Version Compatibility
- Versions and Dependencies
- How to: Configure an app to support .NET Framework 4 or later versions
- What's New
- What's Obsolete in the Class Library
- .NET Framework official support policy ↗
- .NET Framework 4 migration issues

Application compatibility in .NET Framework

Article • 08/10/2023

Compatibility is an important goal of each .NET Framework release. Compatibility ensures that each version is additive, so previous versions will continue to work. On the other hand, changes to previous functionality (for example, to improve performance, address security issues, or fix bugs) can cause compatibility problems in existing code or existing applications that run under a later version.

Each app targets a specific version of .NET Framework by:

- Defining a target framework in Visual Studio.
- Specifying the target framework in a project file.
- Applying a [TargetFrameworkAttribute](#) to the source code.

When migrating from one version of .NET Framework to another, there are two types of changes to consider:

- [Runtime changes](#)
- [Retargeting changes](#)

Runtime changes

Runtime issues are those that arise when a new runtime is placed on a machine and an app's behavior changes. When running on a newer version than what was targeted, .NET Framework uses *quirked* behavior to mimic the older targeted version. The app runs on the newer version but acts as if it's running on the older version. Many of the compatibility issues between versions of .NET Framework are mitigated through this quirk model. For example, if a binary was compiled for .NET Framework 4.0 but runs on a machine with .NET Framework 4.5 or later, it runs in .NET Framework 4.0 compatibility mode. This means that many of the changes in the later version don't affect the binary.

The version of .NET Framework that an application targets is determined by the target version of the entry assembly for the application domain that the code runs in. All additional assemblies loaded in that application domain target that version. For example, in the case of an executable, the version that the executable targets is the compatibility mode all assemblies in that application domain run under.

Retargeting changes

Retargeting changes are those that arise when an assembly is recompiled to target a newer version. Targeting a newer version means the assembly opts into the new features as well as potential compatibility issues for old features.

Impact classification

In the articles that describe runtime and retargeting changes, for example, [Retargeting changes for migration to .NET Framework 4.8.x](#), individual items are classified by their expected impact as follows:

Major

A significant change that affects a large number of apps or that requires substantial modification of code.

Minor

A change that affects a small number of apps or that requires minor modification of code.

Edge case

A change that affects apps under very specific scenarios that are not common.

Transparent

A change that has no noticeable effect on the app's developer or user. The app should not require modification because of this change.

See also

- [Versions and dependencies](#)
- [What's new](#)
- [What's obsolete](#)

Runtime changes for migration to .NET Framework 4.5.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.5](#), [4.5.1](#), and [4.5.2](#).

.NET Framework 4.5

ASP.NET

GridViews with AllowCustomPaging set to true may fire thePageIndexChanging event when leaving the final page of the view

Details

A bug in the .NET Framework 4.5 causes [System.Web.UI.WebControls.GridView.OnPageIndexChanging](#) to sometimes not fire for [System.Web.UI.WebControls.GridViews](#) that have enabled [System.Web.UI.WebControls.GridView.AllowCustomPaging](#).

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework. As a work-around, the app can do an explicit BindGrid on any `Page_Load` that would hit these conditions (the [System.Web.UI.WebControls.GridView](#) is on the last page and [LastSystem.Web.UI.WebControls.GridView.PageSize](#) is different from [System.Web.UI.WebControls.GridView.PageSize](#)). Alternatively, the app can be modified to allow paging (instead of custom paging), as that scenario does not demonstrate the problem.

Name	Value
Scope	Minor
Version	4.5

Name	Value
Type	Runtime

Affected APIs

- [GridView.AllowCustomPaging](#)

HttpRequest.ContentEncoding property prohibits UTF7

Details

Beginning in .NET Framework 4.5, UTF-7 encoding is prohibited in [System.Web.HttpRequests](#)' bodies. Data for applications that depend on incoming UTF-7 data will not decode properly in some cases.

Suggestion

Ideally, applications should be updated to not use UTF-7 encoding in [System.Web.HttpRequests](#). Alternatively, legacy behavior can be restored by using the `aspnet:AllowUtf7RequestContentEncoding` attribute of the [appSettings](#) element.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [HttpRequest.ContentEncoding](#)

HttpUtility.JavaScriptStringEncode escapes ampersand

Details

Starting with the .NET Framework 4.5, [System.Web.HttpUtility.JavaScriptStringEncode\(String\)](#) escapes the ampersand (&) character.

Suggestion

If your app depends on the previous behavior of this method, you can add an `aspnet:JavaScriptDoNotEncodeAmpersand` setting to the [ASP.NET appSettings element](#) in your configuration file.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `HttpUtility.JavaScriptStringEncode(String)`
- `HttpUtility.JavaScriptStringEncode(String, Boolean)`

iPad should not be used in custom capabilities file because it is now a browser capability

Details

Beginning in .NET Framework 4.5, iPad is an identifier in the default ASP.NET browser capabilities file, so it should not be used in a custom capabilities file

Suggestion

If iPad-specific capabilities are required, it is necessary to modify iPad behavior by setting capabilities on the pre-defined gateway refID "IPad" instead of by generating a new "IPad" ID by user agent matching.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Page.LoadComplete event no longer causes System.Web.UI.WebControls.EntityDataSource control to invoke data binding

Details

The [LoadComplete](#) event no longer causes the [System.Web.UI.WebControls.EntityDataSource](#) control to invoke data binding for changes to create/update/delete parameters. This change eliminates an extraneous trip to the database, prevents the values of controls from being reset, and produces behavior that is consistent with other data controls, such as [System.Web.UI.WebControls.SqlDataSource](#) and [System.Web.UI.WebControls.ObjectDataSource](#). This change produces different behavior in the unlikely event that applications rely on invoking data binding in the [LoadComplete](#) event.

Suggestion

If there is a need for databinding, manually invoke databind in an event that is earlier in the post-back.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Profiling ASP.NET MVC4 apps can lead to Fatal Execution Engine Error

Details

Profilers using NGEN /Profile assemblies may crash profiled ASP.NET MVC4 applications on startup with a 'Fatal Execution Engine Exception'

Suggestion

This issue is fixed in the .NET Framework 4.5.2. Alternatively, the profiler may avoid this issue by specifying `COR_PRF_DISABLE_ALL_NGEN_IMAGES` in its event mask.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Sharing session state with ASP.NET StateServer requires all servers in the web farm to use the same .NET Framework version

Details

When enabling [System.Web.SessionState.SessionStateMode.StateServer](#) session state, all of the servers in the given web farm must use the same version of the .NET Framework in order for state to be properly shared.

Suggestion

Be sure to upgrade .NET Framework versions on web servers that share state at the same time.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- `SessionStateMode.StateServer`

WebUtility.HtmlDecode no longer decodes invalid input sequences

Details

By default, decoding methods no longer decode an invalid input sequence into an invalid UTF-16 string. Instead, they return the original input.

Suggestion

The change in decoder output should matter only if you store binary data instead of UTF-16 data in strings. To explicitly control this behavior, set the

`aspnet:AllowRelaxedUnicodeDecoding` attribute of the `appSettings` element to `true` to enable legacy behavior or to `false` to enable the current behavior.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `WebUtility.HtmlDecode(String)`
- `WebUtility.HtmlDecode(String, TextWriter)`
- `WebUtility.UrlDecode(String)`

Core

Assemblies compiled with Regex.CompileToAssembly breaks between 4.0 and 4.5

Details

If an assembly of compiled regular expressions is built with the .NET Framework 4.5 but targets the .NET Framework 4, attempting to use one of the regular expressions in that

assembly on a system with .NET Framework 4 installed throws an exception.

Suggestion

To work around this problem, you can do either of the following:

- Build the assembly that contains the regular expressions with the .NET Framework 4.
- Use an interpreted regular expression.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `Regex.CompileToAssembly(RegexCompilationInfo[], AssemblyName)`
- `Regex.CompileToAssembly(RegexCompilationInfo[], AssemblyName, CustomAttributeBuilder[])`
- `Regex.CompileToAssembly(RegexCompilationInfo[], AssemblyName, CustomAttributeBuilder[], String)`

BlockingCollection<T>.TryTakeFromAny does not throw anymore

Details

If one of the input collections is marked completed,

`TryTakeFromAny(BlockingCollection<T>[], T)` no longer returns -1 and

`TakeFromAny(BlockingCollection<T>[], T)` no longer throws an exception. This change makes it possible to work with collections when one of the collections is either empty or completed, but the other collection still has items that can be retrieved.

Suggestion

If `TryTakeFromAny` returning -1 or `TakeFromAny` throwing were used for control-flow purposes in cases of a blocking collection being completed, such code should now be

changed to use `.Any(b => b.IsCompleted)` to detect that condition.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `BlockingCollection<T>.TakeFromAny(BlockingCollection<T>[], T)`
- `BlockingCollection<T>.TakeFromAny(BlockingCollection<T>[], T, CancellationToken)`
- `BlockingCollection<T>.TryTakeFromAny(BlockingCollection<T>[], T)`
- `BlockingCollection<T>.TryTakeFromAny(BlockingCollection<T>[], T, Int32)`
- `BlockingCollection<T>.TryTakeFromAny(BlockingCollection<T>[], T, TimeSpan)`
- `BlockingCollection<T>.TryTakeFromAny(BlockingCollection<T>[], T, TimeSpan)`

Change in behavior for Task.WaitAll methods with time-out arguments

Details

`Task.WaitAll` behavior was made more consistent in .NET Framework 4.5. In the .NET Framework 4, these methods behaved inconsistently. When the time-out expired, if one or more tasks were completed or canceled before the method call, the method threw an `System.AggregateException` exception. When the time-out expired, if no tasks were completed or canceled before the method call, but one or more tasks entered these states after the method call, the method returned false.

In the .NET Framework 4.5, these method overloads now return false if any tasks are still running when the time-out interval expired, and they throw an `System.AggregateException` exception only if an input task was cancelled (regardless of whether it was before or after the method call) and no other tasks are still running.

Suggestion

If an `System.AggregateException` was being caught as a means of detecting a task that was cancelled prior to the `WaitAll` call being invoked, that code should instead do the

same detection via the `IsCanceled` property (for example: `.Any(t => t.IsCanceled)`) since .NET Framework 4.6 will only throw in that case if all awaited tasks are completed prior to the timeout.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [Task.WaitAll\(Task\[\], Int32\)](#)
- [Task.WaitAll\(Task\[\], Int32, CancellationToken\)](#)
- [Task.WaitAll\(Task\[\], TimeSpan\)](#)

Compiler support for type forwarding when multi-targeting mscorlib

Details

A new CodeDOM feature allows a compiler to compile against the targeted version of mscorlib.dll instead of the .NET Framework 4.5 version of mscorlib.dll.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

ConcurrentQueue<T>.TryPeek can return an erroneous null via its out parameter

Details

In some multi-threaded scenarios,

`System.Collections.Concurrent.ConcurrentQueue<T>.TryPeek(T)` can return true, but populate the out parameter with a null value (instead of the correct, peeked value).

Suggestion

This issue is fixed in the .NET Framework 4.5.1. Upgrading to that Framework will solve the issue.

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

- `ConcurrentQueue<T>.TryPeek(T)`

ETW EventListeners do not capture events from providers with explicit keywords (like the TPL provider)

Details

ETW EventListeners with a blank keyword mask do not properly capture events from providers with explicit keywords. In the .NET Framework 4.5, the TPL provider began providing explicit keywords and triggered this issue. In the .NET Framework 4.6, EventListeners have been updated to no longer have this issue.

Suggestion

To work around this problem, replace calls to `EnableEvents(EventSource, EventLevel)` with calls to the `EnableEvents` overload that explicitly specifies the "any keywords" mask to use: `EnableEvents(eventSource, level, unchecked((EventKeywords)0xFFFFFFFFFFFFFF))`.

Alternatively, this issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [EventListener.EnableEvents\(EventSource, EventLevel\)](#)

Exceptions during unobserved processing in System.Threading.Tasks.Task no longer propagate on finalizer thread

Details

Because the [System.Threading.Tasks.Task](#) class represents an asynchronous operation, it catches all non-severe exceptions that occur during asynchronous processing. In the .NET Framework 4.5, if an exception is not observed and your code never waits on the task, the exception will no longer propagate on the finalizer thread and crash the process during garbage collection. This change enhances the reliability of applications that use the Task class to perform unobserved asynchronous processing.

Suggestion

If an app depends on unobserved asynchronous exceptions propagating to the finalizer thread, the previous behavior can be restored by providing an appropriate handler for the [UnobservedTaskException](#) event, or by setting a [runtime configuration element](#).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [Task.Run\(Action\)](#)

- [Task.Run\(Action, CancellationToken\)](#)
- [Task.Run\(Func<Task>\)](#)
- [Task.Run\(Func<Task>, CancellationToken\)](#)
- [Task.Run<TResult>\(Func<TResult>\)](#)
- [Task.Run<TResult>\(Func<TResult>, CancellationToken\)](#)
- [Task.Run<TResult>\(Func<Task<TResult>>\)](#)
- [Task.Run<TResult>\(Func<Task<TResult>>, CancellationToken\)](#)
- [Task.Start\(\)](#)
- [Task.Start\(TaskScheduler\)](#)

List.Sort algorithm changed

Details

Beginning in .NET Framework 4.5, [System.Collections.Generic.List<T>](#)'s sort algorithm has changed (to be an introspective sort instead of a quick sort).

[System.Collections.Generic.List<T>](#)'s sort has never been stable, but this change may cause different scenarios to sort in unstable ways. That simply means that equivalent items may sort in different orders in subsequent calls of the API.

Suggestion

Because the old sort algorithm was also unstable (though in slightly different ways), there should be no code that depends on equivalent items always sorting in a particular order. If there are instances of code depending upon that and being lucky with the old behavior, that code should be updated to use a comparer that will deterministically sort the items in the desired order.

Name	Value
Scope	Transparent
Version	4.5
Type	Runtime

Affected APIs

- [List<T>.Sort\(\)](#)
- [List<T>.Sort\(IComparer<T>\)](#)
- [List<T>.Sort\(Comparison<T>\)](#)

- `List<T>.Sort(Int32, Int32, IComparer<T>)`

Missing Target Framework Moniker results in 4.0 behavior

Details

Applications without a [System.Runtime.Versioning.TargetFrameworkAttribute](#) applied at the assembly level will automatically run using the semantics (quirks) of the .NET Framework 4.0. To ensure high quality, it is recommended that all binaries be explicitly attributed with a [System.Runtime.Versioning.TargetFrameworkAttribute](#) indicating the version of the .NET Framework they were built with. Note that using a target framework moniker in a project file will cause MSBuild to automatically apply a [System.Runtime.Versioning.TargetFrameworkAttribute](#).

Suggestion

A [System.Runtime.Versioning.TargetFrameworkAttribute](#) should be supplied, either through adding the attribute directly to the assembly or by specifying a target framework in the [project file](#) or through Visual Studio's project properties [GUI ↗](#).

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Some .NET APIs cause first chance (handled) `EntryPointNotFoundExceptions`

Details

In the .NET Framework 4.5, a small number of .NET methods began throwing first chance [System.EntryPointNotFoundExceptions](#). These exceptions were handled within the .NET Framework, but could break test automation that did not expect the first chance

exceptions. These same APIs break some ApiVerifier scenarios when HighVersionLie is enabled.

Suggestion

This bug can be avoided by upgrading to .NET Framework 4.5.1. Alternatively, test automation can be updated to not break on first-chance [System.EntryPointNotFoundExceptions](#).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [Debug.Assert\(Boolean\)](#)
- [Debug.Assert\(Boolean, String\)](#)
- [Debug.Assert\(Boolean, String, String\)](#)
- [Debug.Assert\(Boolean, String, String, Object\[\]\)](#)
- [XmlSerializer\(Type\)](#)

System.Threading.Tasks.Task no longer throw ObjectDisposedException after object is disposed

Details

Except for [IAsyncResult.AsyncWaitHandle](#), [System.Threading.Tasks.Task](#) methods no longer throw an [System.ObjectDisposedException](#) exception after the object is disposed. This change supports the use of cached tasks. For example, a method can return a cached task to represent an already completed operation instead of allocating a new task. This was impossible in previous .NET Framework versions, because any consumer of the task could dispose of it, which rendered it unusable.

Suggestion

Be aware that Task methods may no longer throw [System.ObjectDisposedException](#) in cases when the object is disposed. If an app was depending on this exception to know

that a task was disposed, it should be updated to explicitly check the task's status using [Status](#).

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

System.Uri escaping now supports RFC 3986

Details

URI escaping has changed in .NET Framework 4.5 to support [RFC 3986](#). Specific changes include:

- [System.Uri.EscapeDataString\(String\)](#) escapes reserved characters based on RFC 3986.
- [System.Uri.EscapeUriString\(String\)](#) does not escape reserved characters.
- [System.Uri.UnescapeDataString\(String\)](#) does not throw an exception if it encounters an invalid escape sequence.
- Unreserved escaped characters are un-escaped.

Suggestion

- Update applications to not rely on [System.Uri.UnescapeDataString\(String\)](#) to throw in the case of an invalid escape sequence. Such sequences must be detected directly now.
- Similarly, expect that Escaped and Unescaped URI and Data strings may vary from .NET Framework 4.0 and .NET Framework 4.5 and should not be compared across .NET versions directly. Instead, they should be parsed and normalized in a single .NET version before any comparisons are made.

Name	Value
Scope	Minor

Name	Value
Version	4.5
Type	Runtime

Affected APIs

- [Uri.EscapeDataString\(String\)](#)
- [Uri.EscapeUriString\(String\)](#)
- [Uri.UnescapeDataString\(String\)](#)

Data

Sql_variant data uses sql_variant collation rather than database collation

Details

`sql_variant` data uses `sql_variant` collation rather than database collation.

Suggestion

This change addresses possible data corruption if the database collation differs from the `sql_variant` collation. Applications that rely on the corrupted data may experience failure.

Name	Value
Scope	Transparent
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

SqlBulkCopy uses destination column encoding for strings

Details

When inserting data into a column, [System.Data.SqlClient.SqlBulkCopy](#) uses the encoding of the destination column rather than the default encoding for `VARCHAR` and `CHAR` types. This change eliminates the possibility of data corruption caused by using the default encoding when the destination column does not use the default encoding. In rare cases, an existing application may throw a `SqlException` exception if the change in encoding produces data that is too big to fit into the destination column.

Suggestion

Expect that [System.Data.SqlClient.SqlBulkCopy](#) will no longer corrupt data due to encoding differences. If strings near the destination column's size limit are being copied, it may be necessary to either pre-encode data (to be copied to check that the data will fit in the destination column) or catch [System.Data.SqlClient.SqlExceptions](#).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Data.SqlClient.SqlBulkCopy](#)
- [SqlBulkCopy\(SqlConnection\)](#)

SqlConnection can no longer connect to SQL Server 1997 or databases using the VIA adapter

Details

Connections to SQL Server databases using the [Virtual Interface Adapter \(VIA\)](#) protocol are no longer supported. The protocol used to connect to a SQL Server database is visible in the connection string. A VIA connection will contain via:<servername>. If this

app is connecting to SQL via a protocol other than VIA (tcp: or np: for example), then no breaking change will be encountered. Also, connections to SQL Server 7 (1997) are no longer supported.

Suggestion

The VIA protocol is deprecated, so an alternative protocol should be used to connect to SQL databases. The most common protocol used is TCP/IP. For more information about connecting through TCP/IP, see [Enable the TCP/IP protocol for a database instance](#). If the database is only accessed from within an intranet, the shared pipes protocol may provide better performance if the network is slow.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [SqlConnection\(String\)](#)
- [SqlConnection\(String, SqlCredential\)](#)

SqlConnection.Open fails on Windows 7 with non-IFS Winsock BSP or LSP present

Details

[Open\(\)](#) and [OpenAsync\(CancellationToken\)](#) fail in the .NET Framework 4.5 if running on a Windows 7 machine with a non-IFS Winsock BSP or LSP are present on the computer. To determine whether a non-IFS BSP or LSP is installed, use the `netsh WinSock Show Catalog` command, and examine every `Winsock Catalog Provider Entry` item that is returned. If the Service Flags value has the `0x20000` bit set, the provider uses IFS handles and will work correctly. If the `0x20000` bit is clear (not set), it is a non-IFS BSP or LSP.

Suggestion

This bug has been fixed in the .NET Framework 4.5.2, so it can be avoided by upgrading the .NET Framework. Alternatively, it can be avoided by removing any installed non-IFS

Winsock LSPs.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [SqlConnection.Open\(\)](#)
- [SqlConnection.OpenAsync\(CancellationToken\)](#)

Debugger

Null coalescer values are not visible in debugger until one step later

Details

A bug in the .NET Framework 4.5 causes values set via a null coalescing operation to not be visible in the debugger immediately after the assignment operation is executed when running on the 64-bit version of the Framework.

Suggestion

Stepping one additional time in the debugger will cause the local/field's value to be correctly updated. Also, this issue has been fixed in the .NET Framework 4.6; upgrading to that version of the Framework should solve the issue.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Entity Framework

Change in behavior in Data Definition Language (DDL) APIs

Details

The behavior of DDL APIs when AttachDBFilename is specified has changed as follows:

- Connection strings need not specify an Initial Catalog value. Previously, both AttachDBFilename and Initial Catalog were required.
- If both AttachDBFilename and Initial Catalog are specified and the given MDF file exists, the [DatabaseExists](#) method returns `true`. Previously, it returned `false`.
- If both AttachDBFilename and Initial Catalog are specified and the given MDF file exists, calling the [DeleteDatabase](#) method deletes the files.
- If [DeleteDatabase](#) is called when the connection string specifies an AttachDBFilename value with an MDF that doesn't exist and an Initial Catalog that doesn't exist, the method throws an [InvalidOperationException](#) exception. Previously, it threw a [SqlException](#) exception.

Suggestion

These changes make it easier to build tools and applications that use the DDL APIs. These changes can affect application compatibility in the following scenarios:

- The user writes code that executes a `DROP DATABASE` command directly instead of calling [DeleteDatabase](#) if [DatabaseExists](#) returns `true`. This breaks existing code if the database is not attached but the MDF file exists.
- The user writes code that expects the [DeleteDatabase](#) method to throw a [SqlException](#) rather than an [InvalidOperationException](#) when the Initial Catalog and MDF file don't exist.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Different exception handling for ObjectContext.CreateDatabase and DbProviderServices.CreateDatabase methods

Details

Beginning in .NET Framework 4.5, if database creation fails, `CreateDatabase` methods will attempt to drop the empty database. If that operation succeeds, the original `System.Data.SqlClient.SqlException` will be propagated (instead of the `System.InvalidOperationException` that was always thrown in .NET Framework 4.0)

Suggestion

When catching an `System.InvalidOperationException` while executing `CreateDatabase()` or `CreateDatabase(DbConnection, Nullable<Int32>, StoreItemCollection)`, `SQLExceptions` should now also be caught.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `ObjectContext.CreateDatabase()`
- `DbProviderServices.CreateDatabase(DbConnection, Nullable<Int32>, StoreItemCollection)`

EntityFramework 6.0 loads very slowly in apps launched from Visual Studio

Details

Launching an app from Visual Studio 2013 that uses EntityFramework 6.0 can be very slow.

Suggestion

This issue is fixed in EntityFramework 6.0.2. Update EntityFramework to avoid the performance issue.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

**Log file name created by the
ObjectContext.CreateDatabase method has changed to
match SQL Server specifications**

Details

When the [System.Data.Objects.ObjectContext.CreateDatabase\(\)](#) method is called either directly or by using Code First with the SqlClient provider and an AttachDBFilename value in the connection string, it creates a log file named filename_log.ldf instead of filename.ldf (where filename is the name of the file specified by the AttachDBFilename value). This change improves debugging by providing a log file named according to SQL Server specifications.

Suggestion

If the log file name is important for an app, the app should be updated to expect the standard _log.ldf file name format.

Name	Value
Scope	Edge

Name	Value
Version	4.5
Type	Runtime

Affected APIs

- [ObjectContext.CreateDatabase\(\)](#)

**ObjectContext.Translate and
ObjectContext.ExecuteStoreQuery now support enum type**

Details

In .NET Framework 4.0, the generic parameter `T` of `ObjectContext.Translate` and `ObjectContext.ExecuteStoreQuery` methods could not be an enum. That scenario is now supported.

Suggestion

If `Translate` or `ExecuteStoreQuery` was called on an enum type in .NET Framework 4.0, '0' was returned. If that behavior was desirable, the calls should be replaced with a constant 0 (or the enum equivalent of it).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [ObjectContext.Translate<TElement>\(DbDataReader\)](#)
- [ObjectContext.Translate< TEntity >\(DbDataReader, String, MergeOption\)](#)
- [ObjectContext.ExecuteStoreQuery< TElement >\(String, Object\[\]\)](#)
- [ObjectContext.ExecuteStoreQuery< TEntity >\(String, String, MergeOption, Object\[\]\)](#)

LINQ

Enumerable.Empty<TResult> always returns cached instance

Details

Beginning in .NET Framework 4.5, `Empty<TResult>()` always returns a cached internal instance `IEnumerable<T>`. Previously, `Empty<TResult>()` would cache an empty `IEnumerable<T>` at the time the API was called, meaning that in some conditions in which `Empty<TResult>()` was called rapidly and concurrently, different instances of the type could be returned for different calls to the API.

Suggestion

Because the previous behavior was non-deterministic, code is unlikely to depend on it. However, in the unlikely case that empty enumerables are being compared and expected to sometimes be unequal, explicit empty arrays should be created (`new T[0]`) instead of using `Empty<TResult>()`.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- `Enumerable.Empty<TResult>()`

Managed Extensibility Framework (MEF)

MEF catalogs implement `IEnumerable` and therefore can no longer be used to create a serializer

Details

Starting with the .NET Framework 4.5, MEF catalogs implement `IEnumerable` and therefore can no longer be used to create a serializer (`System.Xml.Serialization.XmlSerializer` object). Trying to serialize a MEF catalog throws an exception.

Suggestion

Can no longer use MEF to create a serializer

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Networking

Deserialization of MailMessage objects serialized under the .NET Framework 4.5 may fail

Details

Starting with the .NET Framework 4.5, `MailMessage` objects can include non-ASCII characters. In the .NET Framework 4, only ASCII characters are supported. `MailMessage` objects that contain non-ASCII characters and that are serialized under the .NET Framework 4.5 or later cannot be deserialized under the .NET Framework 4.

Suggestion

Ensure that your code provides exception handling when deserializing a `MailMessage` object.

Name	Value
Scope	Minor

Name	Value
Version	4.5
Type	Runtime

Affected APIs

- [System.Web.Mail.MailMessage](#)

System.Net.PeerToPeer.Collaboration unavailable on Windows 8

Details

The System.Net.PeerToPeer.Collaboration namespace is unavailable on Windows 8 or above.

Suggestion

Apps that support Windows 8 or above must be updated to not depend on this namespace or its members.

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

- [System.Net.PeerToPeer.Collaboration](#)

Printing

Data written to PrintSystemJobInfo.JobStream must be in XPS format

Details

The [JobStream](#) property exposes the stream of a print job. The user can send raw data to the underlying operating system printing components by writing to this stream. Starting with the .NET Framework 4.5 on Windows 8 and later versions of the Windows operating system, data written to this stream must be in XPS format as a package stream.

Suggestion

To output print content, you can do either of the following:

- Use the [XpsDocumentWriter](#) class to output print content. This is the recommended alternative.
- Ensure that the data sent to the stream returned by the [JobStream](#) property is in XPS format as a package stream.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [PrintSystemJobInfo.JobStream](#)

Serialization

BinaryFormatter can fail to find type from LoadFrom context

Details

As of .NET Framework 4.5, a number of [System.Xml.Serialization.XmlSerializer](#) changes may cause differences in deserialization when using [System.Runtime.Serialization.Formatters.Binary.BinaryFormatter](#) to deserialize types that had been loaded in the LoadFrom context. These changes are due to the new ways [System.Xml.Serialization.XmlSerializer](#) now loads a type which causes different behavior.

when a [System.Runtime.Serialization.Formatters.Binary.BinaryFormatter](#) attempts to deserialize to that type later on. The default serialization binder does not automatically search the LoadFrom context, although it may have worked in some circumstances based on the old behavior of XmlSerializer. Due to the changes, when a type is being loaded from an assembly loaded in a different context, a [System.IO.FileNotFoundException](#) may be thrown.

⚠ Warning

Binary serialization can be dangerous. For more information, see [BinaryFormatter security guide](#).

Suggestion

If this exception is seen, the `Binder` property of the [System.Runtime.Serialization.Formatters.Binary.BinaryFormatter](#) can be set to a custom binder that will find the correct type.

C#

```
var formatter = new BinaryFormatter { Binder = new TypeFinderBinder() }
```

And then the custom binder:

C#

```
public class TypeFinderBinder : SerializationBinder
{
    private static readonly string s_assemblyName =
        Assembly.GetExecutingAssembly().FullName;

    public override Type BindToType(string assemblyName, string typeName)
    {
        return Type.GetType(String.Format(CultureInfo.InvariantCulture, "{0}, {1}",
            typeName, s_assemblyName));
    }
}
```

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Runtime.Serialization.Formatters.Binary.BinaryFormatter](#)
- [BinaryFormatter.Deserialize\(Stream\)](#)
- [BinaryFormatter.Deserialize\(Stream, HeaderHandler\)](#)

SoapFormatter cannot deserialize Hashtable and similar ordered collection objects

Details

The [System.Runtime.Serialization.Formatters.Soap.SoapFormatter](#) does not guarantee that objects serialized under one .NET Framework version will successfully deserialize under a different version. Specifically, some ordered collections (like [System.Collections.Hashtable](#)) added members between 4.0 and 4.5 such that objects of these types cannot deserialize with .NET Framework 4.0 if they were serialized with .NET Framework 4.5. Note that if the serialized data is both serialized and deserialized with the same .NET Framework version, no issue will occur.

Suggestion

[System.Runtime.Serialization.Formatters.Soap.SoapFormatter](#) serialization should be replaced with [System.Runtime.Serialization.Formatters.Binary.BinaryFormatter](#) serialization or [System.Runtime.Serialization.NetDataContractSerializer](#) to be resilient to .NET Framework changes.

Warning

Binary serialization can be dangerous. For more information, see [BinaryFormatter security guide](#).

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `SoapFormatter.Serialize(Stream, Object)`
- `SoapFormatter.Serialize(Stream, Object, Header[])`
- `SoapFormatter.Deserialize(Stream)`
- `SoapFormatter.Deserialize(Stream, HeaderHandler)`

XmlSerializer fails while serializing a type that hides an accessible member with an inaccessible one

Details

When serializing a derived type, the [System.Xml.Serialization.XmlSerializer](#) can fail if the type contains an inaccessible field or property that hides (via the 'new' keyword) a field or property of the same name that was previously accessible (public, for example) on the base type.

Suggestion

This problem can be solved by making the new, hiding member accessible to the [System.Xml.Serialization.XmlSerializer](#) (by marking it public, for example). Alternatively, the following config setting will revert to 4.0 [System.Xml.Serialization.XmlSerializer](#) behavior, which will fix the problem:

XML

```
<system.xml.serialization>
<xmlSerializer useLegacySerializerGeneration="true" />
</system.xml.serialization>
```

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- `XmlSerializer.Serialize(Stream, Object)`
- `XmlSerializer.Serialize(TextWriter, Object)`
- `XmlSerializer.Serialize(Object, XmlSerializationWriter)`
- `XmlSerializer.Serialize(XmlWriter, Object)`

- `XmlSerializer.Serialize(Stream, Object, XmlSerializerNamespaces)`
- `XmlSerializer.Serialize(TextWriter, Object, XmlSerializerNamespaces)`
- `XmlSerializer.Serialize(XmlWriter, Object, XmlSerializerNamespaces)`
- `XmlSerializer.Serialize(XmlWriter, Object, XmlSerializerNamespaces, String)`
- `XmlSerializer.Serialize(XmlWriter, Object, XmlSerializerNamespaces, String, String)`

Web Applications

Managed browser hosting controls from the .NET Framework 1.1 and 2.0 are blocked

Details

Hosting these controls is blocked in Internet Explorer.

Suggestion

Internet Explorer will fail to launch an application that uses managed browser hosting controls. The previous behavior can be restored by setting the `EnableIEHosting` value of the registry subkey `HKLM/SOFTWARE/MICROSOFT/.NETFramework` to `1` for x86 systems and for 32-bit processes on x64 systems, and by setting the `EnableIEHosting` value of the registry subkey `HKLM/SOFTWARE/Wow6432Node/Microsoft/.NETFramework` to `1` for 64-bit processes on x64 systems.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Communication Foundation (WCF)

Error codes for `maxRequestLength` or `maxReceivedMessageSize` are different

Details

Messages in WCF web services hosted in Internet Information Services (IIS) or ASP.NET Development Server that exceed maxRequestLength (in ASP.NET) or maxReceivedMessageSize (in WCF) have different error codeThe HTTP status code has changed from 400 (Bad Request) to 413 (Request Entity Too Large), and messages that exceed either the maxRequestLength or the maxReceivedMessageSize setting throw a [System.ServiceModel.ProtocolException](#) exception. This includes cases in which the transfer mode is Streamed.

Suggestion

This change facilitates debugging in cases where the message length exceeds the limits allowed by ASP.NET or WCF. You must modify any code that performs processing based on an HTTP 400 status code.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

System.ServiceModel.Web.WebServiceHost object no longer adds a default endpoint

Details

The [WebServiceHost](#) object no longer adds a default endpoint if an explicit endpoint has been added by application code.

Suggestion

If users will expect to be able to connect to a default endpoint and other explicit endpoints have been added to the [System.ServiceModel.Web.WebServiceHost](#), default

endpoints should also be added explicitly (using [System.ServiceModel.ServiceHostBase.AddDefaultEndpoints\(\)](#)).

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [ServiceHost.AddServiceEndpoint\(Type, Binding, String\)](#)
- [ServiceHost.AddServiceEndpoint\(Type, Binding, Uri\)](#)
- [ServiceHost.AddServiceEndpoint\(Type, Binding, String, Uri\)](#)
- [ServiceHost.AddServiceEndpoint\(Type, Binding, Uri, Uri\)](#)
- [ServiceHost.AddServiceEndpoint\(Type, Binding, Uri, Uri\)](#)
- [ServiceHostBase.AddServiceEndpoint\(ServiceEndpoint\)](#)
- [ServiceHostBase.AddServiceEndpoint\(String, Binding, String\)](#)
- [ServiceHostBase.AddServiceEndpoint\(String, Binding, Uri\)](#)
- [ServiceHostBase.AddServiceEndpoint\(String, Binding, String, Uri\)](#)
- [ServiceHostBase.AddServiceEndpoint\(String, Binding, Uri, Uri\)](#)

The Replace method in OData URLs is disabled by default

Details

Beginning in the .NET Framework 4.5, the Replace method in OData URLs is disabled by default. When OData Replace is disabled (now by default), any user requests including replace functions (which are uncommon) will fail.

Suggestion

If the replace method is required (which is uncommon), it can be re-enabled through a config settings

([System.Data.Services.Configuration.DataServicesFeaturesSection.ReplaceFunction](#)).

However, an enabled replace method can open security vulnerabilities and should only be used after careful review.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Data.Services DataService<T>](#)

Windows Forms

PreviewLostKeyboardFocus is called repeatedly if its handler shows a Windows Forms message box

Details

Beginning in the .NET Framework 4.5, calling [MessageBox.Show](#) from a [PreviewLostKeyboardFocus](#) handler will cause the handler to re-fire when the message box is closed, potentially resulting in an infinite loop of message boxes.

Suggestion

There are two options to work around this issue:

- It may be avoided by calling [MessageBox.Show](#) instead of [MessageBox.Show](#).
- It may be avoided by showing the message box from a [LostKeyboardFocus](#) event handler (as opposed to a [System.Windows.UIElement.PreviewLostKeyboardFocus](#) event handler).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- ContentElement.PreviewLostKeyboardFocus
- IInputElement.PreviewLostKeyboardFocus
- UIElement.PreviewLostKeyboardFocus
- UIElement3D.PreviewLostKeyboardFocus

WinForm's CheckForOverflowUnderflow property is now true for System.Drawing

Details

The CheckForOverflowUnderflow property for the System.Drawing.dll assembly is set to true.

Suggestion

Previously when overflows occurred, the result would be silently truncated. Now an [System.OverflowException](#) exception is thrown.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Accessing a WPF DataGrid's selected items from a handler of the DataGrid's UnloadingRow event can cause a NullReferenceException

Details

Due to a bug in the .NET Framework 4.5, event handlers for [DataGrid](#) events involving the removal of a row can cause a [System.NullReferenceException](#) to be thrown if they

access the [DataGrid's System.Windows.Controls.Primitives.Selector.SelectedItem](#) or [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) properties.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [DataGrid.UnloadingRow](#)
- [DataGrid.UnloadingRowDetails](#)

Calling DataGrid.CommitEdit from a CellEditEnding handler drops focus

Details

Calling [CommitEdit\(\)](#) from one of the [System.Windows.Controls.DataGrid](#)'s [System.Windows.Controls.DataGrid.CellEditEnding](#) event handlers causes the [System.Windows.Controls.DataGrid](#) to lose focus.

Suggestion

This bug has been fixed in the .NET Framework 4.5.2, so it can be avoided by upgrading the .NET Framework. Alternatively, it can be avoided by explicitly re-selecting the [System.Windows.Controls.DataGrid](#) after calling [System.Windows.Controls.DataGrid.CommitEdit\(\)](#).

Name	Value
Scope	Edge
Version	4.5

Name	Value
Type	Runtime

Affected APIs

- [DataGrid.CommitEdit\(\)](#)
- [DataGrid.CommitEdit\(DataGridEditingUnit, Boolean\)](#)

Calling Items.Refresh on a WPF ListBox, ListView, or DataGrid with items selected can cause duplicate items to appear in the element

Details

In the .NET Framework 4.5, calling `ListBox.Items.Refresh` from code while items are selected in a [System.Windows.Controls.ListBox](#) can cause the selected items to be duplicated in the list. A similar issue occurs with [System.Windows.Controls.ListView](#) and [System.Windows.Controls.DataGrid](#). This is fixed in the .NET Framework 4.6.

Suggestion

This issue may be worked around by programmatically unselecting items before [System.Windows.Data.CollectionView.Refresh\(\)](#) is called and then re-selecting them after the call is completed. Alternatively, this issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [CollectionView.Refresh\(\)](#)

FlowDocument may show an extra line of text

Details

In some cases, a [FlowDocument](#) element will display an extra line of text when running on the .NET Framework 4.5 compared to how it displayed when run on the .NET Framework 4.0. There are no known cases of the change causing any text to be displayed poorly or illegibly, but it could cause text to appear that previously was omitted from a [FlowDocument](#)'s view.

Suggestion

In some cases, decreasing the display element's `PageHeight` property by one can restore the previous number of displayed lines.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [FlowDocument\(\)](#)
- [FlowDocument\(Block\)](#)
- [FlowDocumentReader\(\)](#)
- [FlowDocumentPageViewer\(\)](#)
- [DocumentPageView\(\)](#)

GlyphRun.ComputeInkBoundingBox() and FormattedText.Extent return different values beginning in .NET Framework 4.5

Details

Improvements were made to [ComputeInkBoundingBox\(\)](#) and [Extent](#) in the .NET Framework 4.5 to address issues where the boxes were too small for the contained glyphs in some cases in the .NET Framework 4.0. As a result of this, some bounding boxes will be larger beginning in the .NET Framework 4.5, resulting in subtle differences in UI layout.

Suggestion

Be aware that some glyph bounding box sizes have increased. These changes will usually improve presentation and hit box testing, but if the older (pre-.NET 4.5) behavior is desired, it can be opted into by adding the following entry to the app.config file:

XML

```
<appsettings>
<add key="IncludeAllInkInBoundingBox" value="false">
</appsettings>
```

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [GlyphRun.ComputeInkBoundingBox\(\)](#)
- [FormattedText.Extent](#)

Intermittently unable to scroll to bottom item in ItemsControls (like ListBox and DataGrid) when using custom DataTemplates

Details

In some instances, a bug in the .NET Framework 4.5 is causing ItemsControls (like [System.Windows.Controls.ListBox](#), [System.Windows.Controls.ComboBox](#), [System.Windows.Controls.DataGrid](#), etc.) to not scroll to their bottom item when using custom DataTemplates. If the scrolling is attempted a second time (after scrolling back up), it will work then.

Suggestion

This issue has been fixed in the .NET Framework 4.5.2 and may be addressed by upgrading to that version (or a later version) of the .NET Framework. Alternatively, users

can still drag scroll bars to the final items in these collections, but may need to try twice to do so successfully.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Items.Clear does not remove duplicates from SelectedItems

Details

Suppose a Selector (with multiple selection enabled) has duplicates in its [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) collection - the same item appears more than once. Removing those items from the data source (e.g. by calling Items.Clear) fails to remove them from [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#); only the first instance is removed. Furthermore, subsequent use of [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) (e.g. SelectedItems.Clear()) can encounter problems such as [System.ArgumentException](#), because [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) contains items that are no longer in the data source.

Suggestion

Upgrade if possible to .NET Framework 4.6.2.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [MultiSelector.SelectedItems](#)

ListBoxItem IsSelected binding issue with ObservableCollection<T>.Move

Details

Calling [Move\(Int32, Int32\)](#) or [MoveItem\(Int32, Int32\)](#) on a collection bound to a [System.Windows.Controls.ListBox](#) with items selected can lead to erratic behavior with future selection or unselection of [System.Windows.Controls.ListBox](#) items.

Suggestion

Calling [System.Collections.ObjectModel.Collection<T>.Remove\(T\)](#) and [System.Collections.ObjectModel.Collection<T>.Insert\(Int32, T\)](#) instead of [Move\(Int32, Int32\)](#) will work around this issue. Alternatively, this issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [ObservableCollection<T>.Move\(Int32, Int32\)](#)
- [ObservableCollection<T>.MoveItem\(Int32, Int32\)](#)

New enum values in WPF's PageRangeSelection

Details

Two new members ([System.Windows.Controls.PageRangeSelection.CurrentPage](#) and [System.Windows.Controls.PageRangeSelection.SelectedPages](#)) have been added to the [System.Windows.Controls.PageRangeSelection](#) enum.

Suggestion

In most cases, these changes won't impact user code. Code that depends on a particular number of elements existing in [GetNames\(Type\)](#) or [GetValues\(Type\)](#) calls on the [System.Windows.Controls.PageRangeSelection](#) type should be modified, though.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Windows.Controls.PageRangeSelection](#)

PreviewLostKeyboardFocus is called repeatedly if its handler shows a Windows Forms message box

Details

Beginning in the .NET Framework 4.5, calling [MessageBox.Show](#) from a [PreviewLostKeyboardFocus](#) handler will cause the handler to re-fire when the message box is closed, potentially resulting in an infinite loop of message boxes.

Suggestion

There are two options to work around this issue:

- It may be avoided by calling [MessageBox.Show](#) instead of [MessageBox.Show](#).
- It may be avoided by showing the message box from a [LostKeyboardFocus](#) event handler (as opposed to a [System.Windows.UIElement.PreviewLostKeyboardFocus](#) event handler).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [ContentElement.PreviewLostKeyboardFocus](#)
- [IInputElement.PreviewLostKeyboardFocus](#)
- [UIElement.PreviewLostKeyboardFocus](#)
- [UIElement3D.PreviewLostKeyboardFocus](#)

Right clicking on a WPF DataGrid row header changes the DataGrid selection

Details

Right-clicking a selected [System.Windows.Controls.DataGrid](#) row header while multiple rows are selected results in the [System.Windows.Controls.DataGrid](#)'s selection changing to only that row.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [DataGrid\(\)](#)

Scrolling a WPF TreeView or grouped ListBox in a VirtualizingStackPanel can cause the application to stop responding

Details

In the .NET Framework v4.5, scrolling a WPF [System.Windows.Controls.TreeView](#) in a virtualized stack panel can cause the application to stop responding if there are margins

in the viewport (between the items in the [System.Windows.Controls.TreeView](#), for example, or on an ItemsPresenter element). Additionally, in some cases, different sized items in the view can cause instability even if there are no margins.

Suggestion

This bug can be avoided by upgrading to .NET Framework 4.5.1. Alternatively, margins can be removed from view collections (like [System.Windows.Controls.TreeViews](#)) within virtualized stack panels if all contained items are the same size.

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

- [VirtualizingStackPanel.SetIsVirtualizing\(DependencyObject, Boolean\)](#)

WPF DataTemplate elements are now visible to UIA

Details

Previously, [System.Windows.DataTemplate](#) elements were invisible to UI Automation. Beginning in 4.5, UI Automation will detect these elements. This is useful in many cases, but can break tests that depend on UIA trees not containing [System.Windows.DataTemplate](#) elements.

Suggestion

UI Automation tests for this app may need updated to account for the UIA tree now including previously invisible [System.Windows.DataTemplate](#) elements. For example, tests that expect some elements to be next to each other may now need to expect previously invisible UIA elements in between. Or tests that rely on certain counts or indexes for UIA elements may need updated with new values.

Name	Value
Scope	Edge

Name	Value
Version	4.5
Type	Runtime

Affected APIs

- [DataTemplate\(\)](#)
- [DataTemplate\(Object\)](#)

WPF DispatcherSynchronizationContext.CreateCopy now returns a new copy instead of the current instance

Details

In the .NET Framework 4, [CreateCopy\(\)](#) returned a reference to the current instance, primarily as a performance optimization. In the .NET Framework 4.5, it returns a new instance which makes it possible for the first time to conclude that equal references indicate the executing thread is in the correct synchronization context. It is unlikely that code that checks the identity of these references will be affected, but because of the change, code that calls [CreateCopy\(\)](#) should be tested as part of migration to the .NET Framework 4.5 or newer.

Suggestion

Be aware that [CreateCopy\(\)](#) will now return a new [System.Threading.SynchronizationContext](#) object. Previously, code that used equivalence of references generated this way was not actually checking whether it was in the proper context, but does when built against .NET Framework 4.5 or later. While unlikely to cause issues, exercising the affected code paths should be enough to determine if this poses any problem.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [DispatcherSynchronizationContext.CreateCopy\(\)](#)

WPF TextBox defaults to undo limit of 100

Details

In .NET Framework 4.5, the default undo limit for a WPF textbox is 100 (as opposed to being unlimited in .NET Framework 4.0)

Suggestion

If an undo limit of 100 is too low, the limit can be set explicitly with [UndoLimit](#)

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Windows.Controls.TextBox](#)

WPF TextBox selected text appears a different color when the text box is inactive

Details

In .NET Framework 4.5, when a WPF text box control is inactive (it doesn't have focus), the selected text inside the box will appear a different color than when the control is active.

Suggestion

The previous (.NET Framework 4.0) behavior may be restored by setting the [AreInactiveSelectionHighlightBrushKeysSupported](#) property to `false`.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Windows.Controls.TextBox](#)

WPF TreeViewItem must be used within a TreeView

Details

A change was introduced in 4.5 that restricts usage of [System.Windows.Controls.TreeViewItem](#) elements outside of a [System.Windows.Controls.TreeView](#). This manifests under the following conditions:

- [System.Windows.Controls.TreeViewItem](#)'s visual parent is not a panel. (A [System.Windows.Controls.TreeViewItem](#) generated for a [System.Windows.Controls.TreeView](#) will have a panel as its parent)
- The [System.Windows.Controls.TreeViewItem](#) is a descendant of a [System.Windows.Controls.VirtualizingStackPanel](#) acting as the "items host" for a list control (ListBox, DataGrid, ListView, etc.). Virtualization doesn't need to be enabled.
- The [System.Windows.Controls.VirtualizingStackPanel](#) is item-scrolling (`ScrollUnit="Item"`).
- Someone calls `VirtualizingStackPanel.MakeVisible(v)` to scroll an element `v` into view. This can be done explicitly, or implicitly in a number of ways; perhaps the most common way is simply clicking on `v` to give it the keyboard focus.
- The visual-parent chain from `v` to the [System.Windows.Controls.VirtualizingStackPanel](#) passes through the [System.Windows.Controls.TreeViewItem](#).

In other words, this is seen when a [System.Windows.Controls.TreeViewItem](#) is used outside of a [System.Windows.Controls.TreeView](#), and the user clicks on a descendant of the [System.Windows.Controls.TreeViewItem](#) to bring it into view. If the [System.Windows.Controls.TreeViewItem](#) has no focusable descendants, you'll never see this issue. An example of a situation where this is hit is when a

[System.Windows.Controls.TreeViewItem](#) is the root of a DataTemplate. When this issue is hit, there is an InvalidCastException that occurs within the WPF framework.

Suggestion

A hotfix will be made available for this.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Workflow Foundation (WF)

System.Activities is now APTCA

Details

The assembly is marked with the [System.Security.AllowPartiallyTrustedCallersAttribute](#) attribute.

Suggestion

Derived classes cannot be marked with the [System.Security.SecurityCriticalAttribute](#).

Previously, derived types had to be marked with the [System.Security.SecurityCriticalAttribute](#). However, this change should have no real impact.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

WF serializes Expressions.Literal<T> DateTimes differently now (breaks custom XAML parsers)

Details

The associated [ValueSerializer](#) object will convert a [System.DateTime](#) or [System.DateTimeOffset](#) object whose Second and [System.DateTime.Millisecond](#) components are non-zero and (for a [System.DateTime](#) value) whose [Kind](#) property is not Unspecified to property element syntax instead of a string. This change allows [System.DateTime](#) and [System.DateTimeOffset](#) values to be round-tripped. Custom XAML parsers that assume that input XAML is in the attribute syntax will not function correctly.

Suggestion

This change allows [System.DateTime](#) and [System.DateTimeOffset](#) values to be round-tripped. Custom XAML parsers that assume that input XAML is in the attribute syntax will not function correctly.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

XML, XSLT

XmlSchemaException now sets line positions properly

Details

If the [SetLineInfo](#) value is passed to the Load method and a validation error occurs, the [LineNumber](#) and [LinePosition](#) properties now contain line information.

Suggestion

Exception-handling code that assumes [LineNumber](#) and [LinePosition](#) will not be set should be updated since these properties will now be set properly when SetLineInfo is used while loading XML.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [LoadOptions.SetLineInfo](#)

XmlTextReader DTD entity expansion is limited to 10,000,000 characters

Details

DTD entity expansion is now limited to 10,000,000 characters. Loading XML files without DTD entity expansion or with limited DTD entity expansion is unaffected. Files with DTD entities that expand to more than 10,000,000 characters fail to load, and now throw an exception.

Suggestion

If the limit of DTD entity expansion is too low 10,000,000, the value can be overridden with the [MaxCharactersFromEntities](#) property. An [System.Xml.XmlReaderSettings](#) with the proper [System.Xml.XmlReaderSettings.MaxCharactersFromEntities](#) value can be passed to `xmlReader.Create` that takes [System.Xml.XmlReaderSettings](#) (ie. [Create\(String, XmlReaderSettings\)](#))

Name	Value
Scope	Edge

Name	Value
Version	4.5
Type	Runtime

Affected APIs

- [System.Xml.XmlTextReader](#)
- [XmlTextReader\(\)](#)
- [XmlTextReader\(Stream\)](#)
- [XmlTextReader\(Stream, XmlNameTable\)](#)
- [XmlTextReader\(Stream, XmlNodeType, XmlParserContext\)](#)
- [XmlTextReader\(TextReader\)](#)
- [XmlTextReader\(TextReader, XmlNameTable\)](#)
- [XmlTextReader\(String\)](#)
- [XmlTextReader\(String, Stream\)](#)
- [XmlTextReader\(String, Stream, XmlNameTable\)](#)
- [XmlTextReader\(String, TextReader\)](#)
- [XmlTextReader\(String, TextReader, XmlNameTable\)](#)
- [XmlTextReader\(String, XmlNameTable\)](#)
- [XmlTextReader\(String, XmlNodeType, XmlParserContext\)](#)
- [XmlTextReader\(XmlNameTable\)](#)

XSLT forward compat now works

Details

In the .NET Framework 4, XSLT 1.0 forward compatibility had the following issues:

- Loading a style sheet failed if its version was set to 2.0 and the parser encountered an unrecognized XSLT 1.0 construct.
- The `xsl:sort` construct failed to sort data if the style sheet version was set to 1.1. In the .NET Framework 4.5, these issues have been fixed, and XSLT 1.0 forward compatibility mode works properly.

Suggestion

Most apps should be unaffected, however data will be sorted differently in some cases now that `xsl:sort` is respected. If `xsl:sort` is used in 1.1 style sheets, confirm that apps were not depending on the unsorted order of data. If apps rely on the 4.0 sorting behavior, remove `xsl:sort` from the style sheet.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [System.Xml.Xsl.XslCompiledTransform](#)

XSLT style sheet exception message changed

Details

In the .NET Framework 4.5, the text of the error message when an XSLT file is too complex is "The style sheet is too complex." In previous versions, the error message was "XSLT compile error." Application code that depends on the text of the error message will no longer work. However, the exception types remain the same, so this change should have no real impact.

Suggestion

Update any app code depending on the exception message from this error condition to expect the new message, or (even better) update the code to depend only on the exception type ([System.Xml.XsltException](#)), which has not changed.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [XslCompiledTransform.Load\(String\)](#)
- [XslCompiledTransform.Load\(Type\)](#)
- [XslCompiledTransform.Load\(XmlReader\)](#)
- [XslCompiledTransform.Load\(IXPathNavigable\)](#)

- `XslCompiledTransform.Load(MethodInfo, Byte[], Type[])`
- `XslCompiledTransform.Load(String, XsltSettings, XmlResolver)`
- `XslCompiledTransform.Load(XmlReader, XsltSettings, XmlResolver)`
- `XslCompiledTransform.Load(IXPathNavigable, XsltSettings, XmlResolver)`

.NET Framework 4.5.1

ADO.NET

ADO.NET now attempts to automatically reconnect broken SQL connections

Details

Beginning in the .NET Framework 4.5.1, the .NET Framework will attempt to automatically reconnect broken SQL connections. Although this will typically make apps more reliable, there are edge cases in which an app needs to know that the connection was lost so that it can take some action upon reconnection.

Suggestion

If this feature is undesirable due to compatibility concerns, it can be disabled by setting the `System.Data.SqlClient.SqlConnectionStringBuilder.ConnectRetryCount` property of a connection string (or `System.Data.SqlClient.SqlConnectionStringBuilder`) to 0.

Name	Value
Scope	Edge
Version	4.5.1
Type	Runtime

Affected APIs

- `IDbConnection.ConnectionString`
- `SqlConnection.ConnectionString`
- `ConnectionStringSettings.ConnectionString`
- `DbConnection.ConnectionString`
- `DbConnectionStringBuilder.ConnectionString`

- [SqlConnectionStringBuilder\(\)](#)
- [SqlConnectionStringBuilder\(String\)](#)
- [DbConnectionStringBuilder\(\)](#)
- [DbConnectionStringBuilder\(Boolean\)](#)

Core

A ConcurrentDictionary serialized in .NET Framework 4.5 with NetDataContractSerializer cannot be deserialized by .NET Framework 4.5.1 or 4.5.2

Details

Due to internal changes to the type, [ConcurrentDictionary<TKey,TValue>](#) objects that are serialized with the .NET Framework 4.5 using the [System.Runtime.Serialization.NetDataContractSerializer](#) cannot be deserialized in the .NET Framework 4.5.1 or in the .NET Framework 4.5.2. Note that moving in the other direction (serializing with the .NET Framework 4.5.x and deserializing with the .NET Framework 4.5) works. Similarly, all 4.x cross-version serialization works with the .NET Framework 4.6. Serializing and deserializing with a single version of the .NET Framework is not affected.

Suggestion

If it is necessary to serialize and deserialize a [System.Collections.Concurrent.ConcurrentDictionary<TKey,TValue>](#) between the .NET Framework 4.5 and .NET Framework 4.5.1/4.5.2, a different serializer like the [System.Runtime.Serialization.DataContractSerializer](#) should be used instead of the [System.Runtime.Serialization.NetDataContractSerializer](#). Alternatively, because this issue is addressed in the .NET Framework 4.6, it may be solved by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

ConcurrentQueue<T>.TryPeek can return an erroneous null via its out parameter

Details

In some multi-threaded scenarios,

[System.Collections.Concurrent.ConcurrentQueue<T>.TryPeek\(T\)](#) can return true, but populate the out parameter with a null value (instead of the correct, peeked value).

Suggestion

This issue is fixed in the .NET Framework 4.5.1. Upgrading to that Framework will solve the issue.

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

- [ConcurrentQueue<T>.TryPeek\(T\)](#)

COR_PRF_GC_ROOT_HANDLES are not being enumerated by profilers

Details

In the .NET Framework v4.5.1, the profiling API `RootReferences2()` is incorrectly never returning `COR_PRF_GC_ROOT_HANDLE` (they are returned as `COR_PRF_GC_ROOT_OTHER` instead). This issue is fixed beginning in the .NET Framework 4.6.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Deserialization of objects across appdomains can fail

Details

In some cases, when an app uses two or more app domains with different application bases, trying to deserialize objects in the logical call context across app domains throws an exception.

Suggestion

See [Mitigation: Deserialization of Objects Across App Domains](#)

Name	Value
Scope	Edge
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

EventListener truncates strings with embedded nulls

Details

`System.Diagnostics.Tracing.EventListener` truncates strings with embedded nulls. Null characters are not supported by the `System.Diagnostics.Tracing.EventSource` class. The change only affects apps that use `System.Diagnostics.Tracing.EventListener` to read `System.Diagnostics.Tracing.EventSource` data in process and that use null characters as delimiters.

Suggestion

`System.Diagnostics.Tracing.EventSource` data should be updated, if possible, to not use embedded null characters.

Name	Value
Scope	Edge
Version	4.5.1
Type	Runtime

Affected APIs

- `EventListener()`
- `EventListener.EnableEvents(EventSource, EventLevel)`
- `EventListener.EnableEvents(EventSource, EventLevel, EventKeywords)`
- `EventListener.EnableEvents(EventSource, EventLevel, EventKeywords, IDictionary<String, String>)`

EventSource.WriteEvent impls must pass WriteEvent the same parameters that it received (plus ID)

Details

The runtime now enforces the contract that specifies the following: A class derived from `System.Diagnostics.Tracing.EventSource` that defines an ETW event method must call the base class `EventSource.WriteEvent` method with the event ID followed by the same arguments that the ETW event method was passed.

Suggestion

An `System.IndexOutOfRangeException` exception is thrown if an `System.Diagnostics.Tracing.EventListener` reads `System.Diagnostics.Tracing.EventSource`

data in process for an event source that violates this contract.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Marshal.SizeOf and Marshal.PtrToStructure overloads break dynamic code

Details

Beginning in the .NET Framework 4.5.1, dynamically binding to the methods `SizeOf<T>()`, `SizeOf<T>(T)`, `PtrToStructure(IntPtr, Object)`, `PtrToStructure(IntPtr, Type)`, `PtrToStructure<T>(IntPtr)`, or `PtrToStructure<T>(IntPtr, T)`, (via Windows PowerShell, IronPython, or the C# dynamic keyword, for example) can result in `MethodInvocationExceptions` because new overloads of these methods have been added that may be ambiguous to the scripting engines.

Suggestion

Update scripts to clearly indicate which overload should be used. This can typically done by explicitly casting the methods' type parameters as `Type`. See [this link](#) for more detail and examples of how to workaround the issue.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Some .NET APIs cause first chance (handled) EntryPointNotFoundExceptions

Details

In the .NET Framework 4.5, a small number of .NET methods began throwing first chance [System.EntryPointNotFoundExceptions](#). These exceptions were handled within the .NET Framework, but could break test automation that did not expect the first chance exceptions. These same APIs break some ApiVerifier scenarios when HighVersionLie is enabled.

Suggestion

This bug can be avoided by upgrading to .NET Framework 4.5.1. Alternatively, test automation can be updated to not break on first-chance [System.EntryPointNotFoundExceptions](#).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [Debug.Assert\(Boolean\)](#)
- [Debug.Assert\(Boolean, String\)](#)
- [Debug.Assert\(Boolean, String, String\)](#)
- [Debug.Assert\(Boolean, String, String, Object\[\]\)](#)
- [XmlSerializer\(Type\)](#)

WinRT stream adapters no longer call FlushAsync automatically on close

Details

In Windows Store apps, Windows Runtime stream adapters no longer call the `FlushAsync` method from the `Dispose` method.

Suggestion

This change should be transparent. Developers can restore the previous behavior by writing code like this:

C#

```
using (var stream = GetWindowsRuntimeStream() as Stream)
{
    // do something
    await stream.FlushAsync();
}
```

Name	Value
Scope	Transparent
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Data

ADO.NET now attempts to automatically reconnect broken SQL connections

Details

Beginning in the .NET Framework 4.5.1, the .NET Framework will attempt to automatically reconnect broken SQL connections. Although this will typically make apps more reliable, there are edge cases in which an app needs to know that the connection was lost so that it can take some action upon reconnection.

Suggestion

If this feature is undesirable due to compatibility concerns, it can be disabled by setting the [System.Data.SqlClient.SqlConnectionStringBuilder.ConnectRetryCount](#) property of a connection string (or [System.Data.SqlClient.SqlConnectionStringBuilder](#)) to 0.

Name	Value
Scope	Edge
Version	4.5.1
Type	Runtime

Affected APIs

- [IDbConnection.ConnectionString](#)
- [SqlConnection.ConnectionString](#)
- [ConnectionStringSettings.ConnectionString](#)
- [DbConnection.ConnectionString](#)
- [DbConnectionStringBuilder.ConnectionString](#)
- [SqlConnectionStringBuilder\(\)](#)
- [SqlConnectionStringBuilder\(String\)](#)
- [DbConnectionStringBuilder\(\)](#)
- [DbConnectionStringBuilder\(Boolean\)](#)

Serialization

NetDataContractSerializer fails to deserialize a ConcurrentDictionary serialized with a different .NET version

Details

By design, the [System.Runtime.Serialization.NetDataContractSerializer](#) can be used only if both the serializing and deserializing ends share the same CLR types. Therefore, it is not guaranteed that an object serialized with one version of the .NET Framework can be deserialized by a different

version. [System.Collections.Concurrent.ConcurrentDictionary<TKey,TValue>](#) is a type that is known to not to deserialize correctly if serialized with the .NET Framework 4.5 or earlier and deserialized with the .NET Framework 4.5.1 or later.

Suggestion

There are a number of possible work-arounds for this issue:

- Upgrade the serializing computer to use the .NET Framework 4.5.1, as well.
- Use [System.Runtime.Serialization.DataContractSerializer](#) instead of [System.Runtime.Serialization.NetDataContractSerializer](#) as this does not expect the exact same CLR types at both serializing and deserializing ends.
- Use [System.Collections.Generic.Dictionary< TKey, TValue >](#) instead of [System.Collections.Concurrent.ConcurrentDictionary< TKey, TValue >](#) since it does not exhibit this particular 4.5->4.5.1 break.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

- [NetDataContractSerializer.Deserialize\(Stream\)](#)

Windows Communication Foundation (WCF)

MinFreeMemoryPercentageToActiveService is now respected

Details

This setting establishes the minimum memory that must be available on the server before a WCF service can be activated. It is designed to prevent [System.OutOfMemoryException](#) exceptions. In the .NET Framework 4.5, this setting had no effect. In the .NET Framework 4.5.1, the setting is observed.

Suggestion

An exception occurs if the free memory available on the web server is less than the percentage defined by the configuration setting. Some WCF services that successfully started and ran in a constrained memory environment may now fail.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Scrolling a WPF TreeView or grouped ListBox in a VirtualizingStackPanel can cause the application to stop responding

Details

In the .NET Framework v4.5, scrolling a WPF [System.Windows.Controls.TreeView](#) in a virtualized stack panel can cause the application to stop responding if there are margins in the viewport (between the items in the [System.Windows.Controls.TreeView](#), for example, or on an ItemsPresenter element). Additionally, in some cases, different sized items in the view can cause instability even if there are no margins.

Suggestion

This bug can be avoided by upgrading to .NET Framework 4.5.1. Alternatively, margins can be removed from view collections (like [System.Windows.Controls.TreeViews](#)) within virtualized stack panels if all contained items are the same size.

Name	Value
Scope	Major
Version	4.5
Type	Runtime

Affected APIs

- `VirtualizingStackPanel.SetIsVirtualizing(DependencyObject, Boolean)`

.NET Framework 4.5.2

ASP.NET

ASP.NET MVC now escapes spaces in strings passed in via route parameters

Details

In order to conform to RFC 2396, spaces in route paths are now escaped when populating action parameters from a route. So, whereas `/controller/action/some data` would previously match the route `/controller/action/{data}` and provide `some data` as the data parameter, it will now provide `some%20data` instead.

Suggestion

Code should be updated to unescape string parameters from a route. If the original URI is needed, it can be accessed with the `RequestUri.OriginalString` API.

Name	Value
Scope	Minor
Version	4.5.2
Type	Runtime

Affected APIs

- `RouteAttribute(String)`

No longer able to set `EnableViewStateMac` to false

Details

ASP.NET no longer allows developers to specify `<pages enableViewStateMac="false"/>` or `<%@Page`

`EnableViewStateMac="false" %>`. The view state message authentication code (MAC) is now enforced for all requests with embedded view state. Only apps that explicitly set the `EnableViewStateMac` property to `false` are affected.

Suggestion

`EnableViewStateMac` must be assumed to be true, and any resulting MAC errors must be resolved (as explained in [this guidance](#), which contains multiple resolutions depending on the specifics of what is causing MAC errors).

Name	Value
Scope	Major
Version	4.5.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Profiling ASP.NET MVC4 apps can lead to Fatal Execution Engine Error

Details

Profilers using NGEN /Profile assemblies may crash profiled ASP.NET MVC4 applications on startup with a 'Fatal Execution Engine Exception'

Suggestion

This issue is fixed in the .NET Framework 4.5.2. Alternatively, the profiler may avoid this issue by specifying `COR_PRF_DISABLE_ALL_NGEN_IMAGES` in its event mask.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Data

SqlConnection.Open fails on Windows 7 with non-IFS Winsock BSP or LSP present

Details

[Open\(\)](#) and [OpenAsync\(CancellationToken\)](#) fail in the .NET Framework 4.5 if running on a Windows 7 machine with a non-IFS Winsock BSP or LSP are present on the computer. To determine whether a non-IFS BSP or LSP is installed, use the `netsh WinSock Show Catalog` command, and examine every `Winsock Catalog Provider Entry` item that is returned. If the Service Flags value has the `0x20000` bit set, the provider uses IFS handles and will work correctly. If the `0x20000` bit is clear (not set), it is a non-IFS BSP or LSP.

Suggestion

This bug has been fixed in the .NET Framework 4.5.2, so it can be avoided by upgrading the .NET Framework. Alternatively, it can be avoided by removing any installed non-IFS Winsock LSPs.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [SqlConnection.Open\(\)](#)
- [SqlConnection.OpenAsync\(CancellationToken\)](#)

Entity Framework

EF no longer throws for QueryViews with specific characteristics

Details

Entity Framework no longer throws a [System.StackOverflowException](#) exception when an app executes a query that involves a QueryView with a 0..1 navigation property that attempts to include the related entities as part of the query. For example, by calling
`.Include(e => e.RelatedNavProp).`

Suggestion

This change only affects code that uses QueryViews with 1-0..1 relationships when running queries that call `.Include`. It improves reliability and should be transparent to almost all apps. However, if it causes unexpected behavior, you can disable it by adding the following entry to the `<appSettings>` section of the app's configuration file:

XML

```
<add key="EntityFramework_SimplifyUserSpecifiedViews" value="false" />
```

Name	Value
Scope	Edge
Version	4.5.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Opt-in break to revert from different 4.5 SQL generation to simpler 4.0 SQL generation

Details

Queries that produce JOIN statements and contain a call to a limiting operation without first using `OrderBy` now produce simpler SQL. After upgrading to .NET Framework 4.5,

these queries produced more complicated SQL than previous versions.

Suggestion

This feature is disabled by default. If Entity Framework generates extra JOIN statements that cause performance degradation, you can enable this feature by adding the following entry to the `<appSettings>` section of the application configuration (app.config) file:

XML

```
<add key="EntityFramework_SimplifyLimitOperations" value="true" />
```

Name	Value
Scope	Transparent
Version	4.5.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Calling `DataGrid.CommitEdit` from a `CellEditEnding` handler drops focus

Details

Calling `CommitEdit()` from one of the `System.Windows.Controls.DataGrid`'s `System.Windows.Controls.DataGrid.CellEditEnding` event handlers causes the `System.Windows.Controls.DataGrid` to lose focus.

Suggestion

This bug has been fixed in the .NET Framework 4.5.2, so it can be avoided by upgrading the .NET Framework. Alternatively, it can be avoided by explicitly re-selecting the

[System.Windows.Controls.DataGrid](#) after calling [System.Windows.Controls.DataGrid.CommitEdit\(\)](#).

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [DataGrid.CommitEdit\(\)](#)
- [DataGrid.CommitEdit\(DataGridEditingUnit, Boolean\)](#)

Intermittently unable to scroll to bottom item in ItemsControls (like ListBox and DataGrid) when using custom DataTemplates

Details

In some instances, a bug in the .NET Framework 4.5 is causing ItemsControls (like [System.Windows.Controls.ListBox](#), [System.Windows.Controls.ComboBox](#), [System.Windows.Controls.DataGrid](#), etc.) to not scroll to their bottom item when using custom DataTemplates. If the scrolling is attempted a second time (after scrolling back up), it will work then.

Suggestion

This issue has been fixed in the .NET Framework 4.5.2 and may be addressed by upgrading to that version (or a later version) of the .NET Framework. Alternatively, users can still drag scroll bars to the final items in these collections, but may need to try twice to do so successfully.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF spawns a wisptis.exe process which can freeze the mouse

Details

An issue was introduced in 4.5.2 that causes `wisptis.exe` to be spawned that can freeze mouse input.

Suggestion

A fix for this issue is available in a servicing release of the .NET Framework 4.5.2 (hotfix rollup 3026376), or by upgrading to the .NET Framework 4.6

Name	Value
Scope	Major
Version	4.5.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

XML

XML parsing changes

Name	Value
Scope	Minor
Version	4.5.2
Type	Runtime

Details

For security reasons, the following changes were introduced into XML parsing APIs:

- `XmlReaderSettings.MaxCharactersFromEntities` is set to 10 million when `XmlReaderSettings` is initialized.
- `XmlReaderSettings.XmlResolver` is set to `null` by default.

ⓘ Note

`XmlReaderSettings` is used by all XML parsers, so while this change helps the `XmlReader` case, it also affects other scenarios.

Suggestion

To revert to the previous behavior, you can set a value in the registry. Add a DWORD value named `EnableLegacyXmlSettings` to the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\XML` registry key, and set its value to `1`. You can also add the registry value in the `HKEY_CURRENT_USER` hive instead.

Affected APIs

- `System.Xml.XmlReaderSettings.MaxCharactersFromEntities`
- `System.Xml.XmlReaderSettings.XmlResolver`

In addition, any XML API that depends on `XmlResolver`, either directly or indirectly, is affected.

Runtime changes for migration to .NET Framework 4.6.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.6](#), [4.6.1](#), and [4.6.2](#).

.NET Framework 4.6

ASP.NET

GridViews with AllowCustomPaging set to true may fire thePageIndexChanging event when leaving the final page of the view

Details

A bug in the .NET Framework 4.5 causes [System.Web.UI.WebControls.GridView.OnPageIndexChanging](#) to sometimes not fire for [System.Web.UI.WebControls.GridViews](#) that have enabled [System.Web.UI.WebControls.GridView.AllowCustomPaging](#).

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework. As a work-around, the app can do an explicit BindGrid on any `Page_Load` that would hit these conditions (the [System.Web.UI.WebControls.GridView](#) is on the last page and [LastSystem.Web.UI.WebControls.GridView.PageSize](#) is different from [System.Web.UI.WebControls.GridView.PageSize](#)). Alternatively, the app can be modified to allow paging (instead of custom paging), as that scenario does not demonstrate the problem.

Name	Value
Scope	Minor
Version	4.5

Name	Value
Type	Runtime

Affected APIs

- [GridView.AllowCustomPaging](#)

Core

A ConcurrentDictionary serialized in .NET Framework 4.5 with NetDataContractSerializer cannot be deserialized by .NET Framework 4.5.1 or 4.5.2

Details

Due to internal changes to the type, [ConcurrentDictionary<TKey,TValue>](#) objects that are serialized with the .NET Framework 4.5 using the [System.Runtime.Serialization.NetDataContractSerializer](#) cannot be deserialized in the .NET Framework 4.5.1 or in the .NET Framework 4.5.2. Note that moving in the other direction (serializing with the .NET Framework 4.5.x and deserializing with the .NET Framework 4.5) works. Similarly, all 4.x cross-version serialization works with the .NET Framework 4.6. Serializing and deserializing with a single version of the .NET Framework is not affected.

Suggestion

If it is necessary to serialize and deserialize a [System.Collections.Concurrent.ConcurrentDictionary<TKey,TValue>](#) between the .NET Framework 4.5 and .NET Framework 4.5.1/4.5.2, a different serializer like the [System.Runtime.Serialization.DataContractSerializer](#) should be used instead of the [System.Runtime.Serialization.NetDataContractSerializer](#). Alternatively, because this issue is addressed in the .NET Framework 4.6, it may be solved by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5.1

Name	Value
Type	Runtime

Affected APIs

Not detectable via API analysis.

AppDomainSetup.DynamicBase is no longer randomized by UseRandomizedStringHashAlgorithm

Details

Prior to the .NET Framework 4.6, the value of [DynamicBase](#) would be randomized between application domains, or between processes, if [UseRandomizedStringHashAlgorithm](#) was enabled in the app's config file. Beginning in the .NET Framework 4.6, [DynamicBase](#) will return a stable result between different instances of an app running, and between different app domains. Dynamic bases will still differ for different apps; this change only removes the random naming element for different instances of the same app.

Suggestion

Be aware that enabling [UseRandomizedStringHashAlgorithm](#) will not result in [DynamicBase](#) being randomized. If a random base is needed, it must be produced in your app's code rather than via this API.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

- [AppDomainSetup.DynamicBase](#)

Calling `Attribute.GetCustomAttributes` on an indexer property no longer throws `AmbiguousMatchException` if the ambiguity can be resolved by index's type

Details

Prior to the .NET Framework 4.6, calling `GetCustomAttribute(s)` on an indexer property which differed from another property only by the type of the index would result in an `System.Reflection.AmbiguousMatchException`. Beginning in the .NET Framework 4.6, the property's attributes will be correctly returned.

Suggestion

Be aware that `GetCustomAttribute(s)` will work more frequently now. If an app was previously relying on the `System.Reflection.AmbiguousMatchException`, reflection should now be used to explicitly look for multiple indexers, instead.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

- `Attribute.GetCustomAttribute(MemberInfo, Type)`
- `Attribute.GetCustomAttribute(MemberInfo, Type, Boolean)`
- `Attribute.GetCustomAttributes(MemberInfo)`
- `Attribute.GetCustomAttributes(MemberInfo, Boolean)`
- `Attribute.GetCustomAttributes(MemberInfo, Type)`
- `Attribute.GetCustomAttributes(MemberInfo, Type, Boolean)`
- `CustomAttributeExtensions.GetCustomAttribute(MemberInfo, Type)`
- `CustomAttributeExtensions.GetCustomAttribute(MemberInfo, Type, Boolean)`
- `CustomAttributeExtensions.GetCustomAttribute<T>(MemberInfo)`
- `CustomAttributeExtensions.GetCustomAttribute<T>(MemberInfo, Boolean)`
- `CustomAttributeExtensions.GetCustomAttributes(MemberInfo)`
- `CustomAttributeExtensions.GetCustomAttributes(MemberInfo, Boolean)`
- `CustomAttributeExtensions.GetCustomAttributes(MemberInfo, Type)`
- `CustomAttributeExtensions.GetCustomAttributes(MemberInfo, Type, Boolean)`

- `CustomAttributeExtensions.GetCustomAttributes<T>(MemberInfo)`
- `CustomAttributeExtensions.GetCustomAttributes<T>(MemberInfo, Boolean)`

COR_PRF_GC_ROOT_HANDL_Es are not being enumerated by profilers

Details

In the .NET Framework v4.5.1, the profiling API `RootReferences2()` is incorrectly never returning `COR_PRF_GC_ROOT_HANDLE` (they are returned as `COR_PRF_GC_ROOT_OTHER` instead). This issue is fixed beginning in the .NET Framework 4.6.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

ETW EventListeners do not capture events from providers with explicit keywords (like the TPL provider)

Details

ETW EventListeners with a blank keyword mask do not properly capture events from providers with explicit keywords. In the .NET Framework 4.5, the TPL provider began providing explicit keywords and triggered this issue. In the .NET Framework 4.6, EventListeners have been updated to no longer have this issue.

Suggestion

To work around this problem, replace calls to `EnableEvents(EventSource, EventLevel)` with calls to the `EnableEvents` overload that explicitly specifies the "any keywords" mask to use: `EnableEvents(eventSource, level, unchecked((EventKeywords)0xFFFFFFFFFFFFFF))`.

Alternatively, this issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- `EventListener.EnableEvents(EventSource, EventLevel)`

Persian calendar now uses the Hijri solar algorithm

Details

Starting with the .NET Framework 4.6, the `System.Globalization.PersianCalendar` class uses the Hijri solar algorithm. Converting dates between the `System.Globalization.PersianCalendar` and other calendars may produce a slightly different result beginning with the .NET Framework 4.6 for dates earlier than 1800 or later than 2023 (Gregorian).Also, `PersianCalendar.MinSupportedDateTime` is now `March 22, 0622` instead of `March 21, 0622`.

Suggestion

Be aware that some early or late dates may be slightly different when using the `PersianCalendar` in .NET Framework 4.6. Also, when serializing dates between processes which may run on different .NET Framework versions, do not store them as `PersianCalendar` date strings (since those values may be different).

Name	Value
Scope	Minor

Name	Value
Version	4.6
Type	Runtime

Affected APIs

- [System.Globalization.PersianCalendar](#)

Reflection objects can no longer be passed from managed code to out-of-process DCOM clients

Details

Reflection objects can no longer be passed from managed code to out-of-process DCOM clients. The following types are affected:

- [System.Reflection.Assembly](#)
- [System.Reflection.MemberInfo](#) (and its derived types, including [System.Reflection.FieldInfo](#), [System.Reflection MethodInfo](#), [System.Type](#), and [System.Reflection.TypeInfo](#))
- [System.Reflection.MethodBody](#)
- [System.Reflection.Module](#)
- [System.Reflection.ParameterInfo](#)

Calls to `IMarshal` for the object return `E_NOINTERFACE`.

Suggestion

Update marshaling code to work with non-reflection objects.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

- [System.Reflection.Assembly](#)
- [System.Reflection.FieldInfo](#)
- [System.Reflection.MemberInfo](#)
- [System.Reflection.MethodBody](#)
- [System.Reflection MethodInfo](#)
- [System.Reflection.Module](#)
- [System.Reflection.ParameterInfo](#)
- [System.Reflection.TypeInfo](#)
- [System.Type](#)

TargetFrameworkName for default app domain no longer defaults to null if not set

Details

The [System.AppDomainSetup.TargetFrameworkName](#) was previously null in the default app domain, unless it was explicitly set. Beginning in 4.6, the [System.AppDomainSetup.TargetFrameworkName](#) property for the default app domain will have a default value derived from the TargetFrameworkAttribute (if one is present). Non-default app domains will continue to inherit their [System.AppDomainSetup.TargetFrameworkName](#) from the default app domain (which will not default to null in 4.6) unless it is explicitly overridden.

Suggestion

Code should be updated to not depend on [TargetFrameworkName](#) defaulting to null. If it is required that this property continue to evaluate to null, it can be explicitly set to that value.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

- [AppDomainSetup.TargetFrameworkName](#)

X509Certificate2.ToString(Boolean) does not throw now when .NET cannot handle the certificate

Details

In .NET Framework 4.5.2 and earlier versions, this method would throw if `true` was passed for the verbose parameter and there were certificates installed that weren't supported by the .NET Framework. Now, the method will succeed and return a valid string that omits the inaccessible portions of the certificate.

Suggestion

Any code depending on [X509Certificate2.ToString\(Boolean\)](#) should be updated to expect that the returned string may exclude some certificate data (such as public key, private key, and extensions) in some cases in which the API would have previously thrown.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

- [X509Certificate2.ToString\(Boolean\)](#)

Data

Attempting a TCP/IP connection to a SQL Server database that resolves to `localhost` fails

Details

In the .NET Framework 4.6 and 4.6.1, attempting a TCP/IP connection to a SQL Server database that resolves to `localhost` fails with the error, "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server

is configured to allow remote connections. (provider: SQL Network Interfaces, error: 26 - Error Locating Server/Instance Specified)"

Suggestion

This issue has been addressed and the previous behavior restored in the .NET Framework 4.6.2. To connect to a SQL Server database that resolves to `localhost`, upgrade to the .NET Framework 4.6.2.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

Debugger

Null coalescer values are not visible in debugger until one step later

Details

A bug in the .NET Framework 4.5 causes values set via a null coalescing operation to not be visible in the debugger immediately after the assignment operation is executed when running on the 64-bit version of the Framework.

Suggestion

Stepping one additional time in the debugger will cause the local/field's value to be correctly updated. Also, this issue has been fixed in the .NET Framework 4.6; upgrading to that version of the Framework should solve the issue.

Name	Value
Scope	Edge

Name	Value
Version	4.5
Type	Runtime

Affected APIs

Not detectable via API analysis.

Networking

ContentDisposition DateTimes returns slightly different string

Details

String representations of [System.Net.Mime.ContentDisposition](#)'s have been updated, beginning in 4.6, to always represent the hour component of a [System.DateTime](#) with two digits. This is to comply with [RFC822](#) and [RFC2822](#). This causes [ToString\(\)](#) to return a slightly different string in 4.6 in scenarios where one of the disposition's time elements was before 10:00 AM. Note that ContentDispositions are sometimes serialized via converting them to strings, so any [ToString\(\)](#) operations, serialization, or [GetHashCode](#) calls should be reviewed.

Suggestion

Do not expect that string representations of ContentDispositions from different .NET Framework versions will correctly compare to one another. Convert the strings back to ContentDispositions, if possible, before conducting a comparison.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

- [ContentDisposition.ToString\(\)](#)
- [ContentDisposition.GetHashCode\(\)](#)

Serialization

Exception message has changed for failed DataContract serialization in case of an unknown type

Details

Beginning in the .NET Framework 4.6, the exception message given if a [System.Runtime.Serialization.DataContractSerializer](#) or [System.Runtime.Serialization.JsonDataContractJsonSerializer](#) fails to serialize or deserialize due to missing 'known types' has been clarified.

Suggestion

Apps should not depend on specific exception messages. If an app depends on this message, either update it to expect the new message or (preferably) change it to depend only on the exception type.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

- [DataContractJsonSerializer\(Type\)](#)
- [DataContractJsonSerializer\(Type, IEnumerable<Type>\)](#)
- [DataContractJsonSerializer\(Type, DataContractJsonSerializerSettings\)](#)
- [DataContractJsonSerializer\(Type, String\)](#)
- [DataContractJsonSerializer\(Type, String, IEnumerable<Type>\)](#)
- [DataContractJsonSerializer\(Type, XmlDictionaryString\)](#)
- [DataContractJsonSerializer\(Type, XmlDictionaryString, IEnumerable<Type>\)](#)
- [DataContractJsonSerializer\(Type, IEnumerable<Type>, Int32, Boolean, IDataContractSurrogate, Boolean\)](#)

- `DataContractJsonSerializer(Type, String, IEnumerable<Type>, Int32, Boolean, IDataContractSurrogate, Boolean)`
- `DataContractJsonSerializer(Type, XmlDictionaryString, IEnumerable<Type>, Int32, Boolean, IDataContractSurrogate, Boolean)`
- `DataContractSerializer(Type)`
- `DataContractSerializer(Type, DataContractSerializerSettings)`
- `DataContractSerializer(Type, IEnumerable<Type>)`
- `DataContractSerializer(Type, String, String)`
- `DataContractSerializer(Type, String, String, IEnumerable<Type>)`
- `DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString)`
- `DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString, IEnumerable<Type>)`
- `DataContractSerializer(Type, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate)`
- `DataContractSerializer(Type, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate, DataContractResolver)`
- `DataContractSerializer(Type, String, String, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate)`
- `DataContractSerializer(Type, String, String, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate, DataContractResolver)`
- `DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate)`
- `DataContractSerializer(Type, XmlDictionaryString, XmlDictionaryString, IEnumerable<Type>, Int32, Boolean, Boolean, IDataContractSurrogate, DataContractResolver)`

Setup and Deployment

Product versioning changes in the .NET Framework 4.6 and later versions

Details

Product versioning has changed from the previous releases of the .NET Framework, and particularly from the .NET Framework 4, 4.5, 4.5.1, and 4.5.2. The following are the detailed changes:

- The value of the `Version` entry in the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` key has changed to `4.6.xxxxx` for the .NET

Framework 4.6 and its point releases, and to `4.7.xxxxx` for the .NET Framework 4.7 and 4.7.1. In the .NET Framework 4.5, 4.5.1, and 4.5.2, it had the format `4.5.xxxxx`.

- The file and product versioning for .NET Framework files has changed from the earlier versioning scheme of 4.0.30319.x to 4.6.X.0 for the .NET Framework 4.6 and its point releases, and to 4.7.X.0 for the .NET Framework 4.7 and 4.7.1. You can see these new values when you view the file's Properties after right-clicking on a file.
- The [AssemblyFileVersionAttribute](#) and [AssemblyInformationalVersionAttribute](#) attributes for managed assemblies have Version values in the form 4.6.X.0 for the .NET Framework 4.6 and its point releases, and 4.7.X.0 for the .NET Framework 4.7 and 4.7.1.
- In the .NET Framework 4.6, 4.6.1, 4.6.2, 4.7, and 4.7.1, the [Environment.Version](#) property returns the fixed version string `4.0.30319.42000`. In the .NET Framework 4, 4.5, 4.5.1, and 4.5.2, it returns version strings in the format `4.0.30319.xxxxx` (for example, "4.0.30319.18010"). Note that we do not recommend application code taking any new dependency on the [Environment.Version](#) property.

For more information, see [How to: Determine which .NET Framework Versions Are Installed](#).

Suggestion

In general, applications should depend on the recommended techniques for detecting such things as the runtime version of the .NET Framework and the installation directory:

- To detect the runtime version of the .NET Framework, see [How to: Determine Which .NET Framework Versions Are Installed](#).
- To determine the installation path for the .NET Framework, use the value of the `InstallPath` entry in the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` key.

Important

The subkey name is `NET Framework Setup`, not `.NET Framework Setup`.

- To determine the directory path to the .NET Framework common language runtime, call the [RuntimeEnvironment.GetRuntimeDirectory\(\)](#) method.
- To get the CLR version, call the [RuntimeEnvironment.GetSystemVersion\(\)](#) method. For the .NET Framework 4 and its point releases (the .NET Framework 4.5, 4.5.1, 4.5.2, and .NET Framework 4.6, 4.6.1, 4.6.2, 4.7, and 4.7.1), it returns the string v4.0.30319.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

The .NET Framework 4.6 does not use a 4.5.x.x version when registering itself in the registry

Details

As one might expect, the version key set in the registry (at

`HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\NET Framework Setup\NDP\v4\Full`) for the .NET Framework 4.6 begins with '4.6', not '4.5'. Apps that depend on these registry keys to know which .NET Framework versions are installed on a machine should be updated to understand that 4.6 is a new possible version, and one that is compatible with previous 4.5.x releases.

Suggestion

Update apps probing for a .NET Framework 4.5 install by looking for 4.5 registry keys to also accept 4.6.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Communication Foundation (WCF)

WCF services that use NETTCP with SSL security and MD5 certificate authentication

Details

The .NET Framework 4.6 adds TLS 1.1 and TLS 1.2 to the WCF SSL default protocol list. When both client and server machines have the .NET Framework 4.6 or later installed, TLS 1.2 is used for negotiation. TLS 1.2 does not support MD5 certificate authentication. As a result, if a customer uses an MD5 certificate, the WCF client will fail to connect to the WCF service.

Suggestion

You can work around this issue so that a WCF client can connect to a WCF server by doing any of the following:

- Update the certificate to not use the MD5 algorithm. This is the recommended solution.
- If the binding is not dynamically configured in source code, update the application's configuration file to use TLS 1.1 or an earlier version of the protocol. This allows you to continue to use a certificate with the MD5 hash algorithm.

Warning

This workaround is not recommended, since a certificate with the MD5 hash algorithm is considered insecure.

The following configuration file does this:

XML

```
<configuration>
  <system.serviceModel>
    <bindings>
      <netTcpBinding>
        <binding>
          <security mode=
            "None/Transport/Message/TransportWithMessageCredential" >
            <transport clientCredentialType="None/Windows/Certificate"
              protectionLevel="None/Sign/EncryptAndSign"
              sslProtocols="Ssl3/Tls1/Tls11">
              </transport>
            </security>
          </binding>
        </netTcpBinding>
      </bindings>
    </system.serviceModel>
</configuration>
```

```
</netTcpBinding>
</bindings>
</system.ServiceModel>
</configuration>
```

- If the binding is dynamically configured in source code, update the [TcpTransportSecurity.SslProtocols](#) property to use TLS 1.1 ([SslProtocols.Tls11](#) or an earlier version of the protocol in the source code).

⚠ Warning

This workaround is not recommended, since a certificate with the MD5 hash algorithm is considered insecure.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Accessing a WPF DataGrid's selected items from a handler of the DataGrid's UnloadingRow event can cause a NullReferenceException

Details

Due to a bug in the .NET Framework 4.5, event handlers for [DataGrid](#) events involving the removal of a row can cause a [System.NullReferenceException](#) to be thrown if they access the [DataGrid](#)'s [System.Windows.Controls.Primitives.Selector.SelectedItem](#) or [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) properties.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [DataGrid.UnloadingRow](#)
- [DataGrid.UnloadingRowDetails](#)

Calling `Items.Refresh` on a WPF ListBox, ListView, or DataGrid with items selected can cause duplicate items to appear in the element

Details

In the .NET Framework 4.5, calling `ListBox.Items.Refresh` from code while items are selected in a [System.Windows.Controls.ListBox](#) can cause the selected items to be duplicated in the list. A similar issue occurs with [System.Windows.Controls.ListView](#) and [System.Windows.Controls.DataGrid](#). This is fixed in the .NET Framework 4.6.

Suggestion

This issue may be worked around by programmatically unselecting items before [System.Windows.DataCollectionView.Refresh\(\)](#) is called and then re-selecting them after the call is completed. Alternatively, this issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [CollectionView.Refresh\(\)](#)

Coercing SelectionBox.Highlighted

Details

Certain sequences of actions involving a [System.Windows.Controls.ComboBox](#) and its data source can result in a [System.NullReferenceException](#).

Suggestion

If possible, upgrade to .NET Framework 4.6.2.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

- [ComboBox.IsSelectionBoxHighlighted](#)

ListBoxItem IsSelected binding issue with ObservableCollection<T>.Move

Details

Calling [Move\(Int32, Int32\)](#) or [MoveItem\(Int32, Int32\)](#) on a collection bound to a [System.Windows.Controls.ListBox](#) with items selected can lead to erratic behavior with future selection or unselection of [System.Windows.Controls.ListBox](#) items.

Suggestion

Calling [System.Collections.ObjectModel.Collection<T>.Remove\(T\)](#) and [System.Collections.ObjectModel.Collection<T>.Insert\(Int32, T\)](#) instead of [Move\(Int32, Int32\)](#) will work around this issue. Alternatively, this issue has been fixed in the .NET

Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- [ObservableCollection<T>.Move\(Int32, Int32\)](#)
- [ObservableCollection<T>.MoveItem\(Int32, Int32\)](#)

Right clicking on a WPF DataGrid row header changes the DataGrid selection

Details

Right-clicking a selected [System.Windows.Controls.DataGrid](#) row header while multiple rows are selected results in the [System.Windows.Controls.DataGrid](#)'s selection changing to only that row.

Suggestion

This issue has been fixed in the .NET Framework 4.6 and may be addressed by upgrading to that version of the .NET Framework.

Name	Value
Scope	Edge
Version	4.5
Type	Runtime

Affected APIs

- [DataGrid\(\)](#)

WPF spawns a wisptis.exe process which can freeze the mouse

Details

An issue was introduced in 4.5.2 that causes `wisptis.exe` to be spawned that can freeze mouse input.

Suggestion

A fix for this issue is available in a servicing release of the .NET Framework 4.5.2 (hotfix rollup 3026376), or by upgrading to the .NET Framework 4.6

Name	Value
Scope	Major
Version	4.5.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF spell checking in text-enabled controls will not work in Windows 10 for languages not in the OS's input language list

Details

When running on Windows 10, the spell checker may not work for WPF text-enabled controls because platform spell-checking capabilities are available only for languages present in the input languages list. In Windows 10, when a language is added to the list of available keyboards, Windows automatically downloads and installs a corresponding Feature on Demand (FOD) package that provides spell-checking capabilities. By adding the language to the input languages list, the spell checker will be supported.

Suggestion

Be aware that the language or text to be spell-checked must be added as an input language for spell-checking to work in Windows 10.

Name	Value
Scope	Edge
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF windows are rendered without clipping when extending outside a single monitor

Details

In the .NET Framework 4.6 running on Windows 8 and above, the entire window is rendered without clipping when it extends outside of single display in a multi-monitor scenario. This is different from previous versions of the .NET Framework which would clip WPF windows that extended beyond a single display.

Suggestion

This behavior (whether to clip or not) can be explicitly set using the `<EnableMultiMonitorDisplayClipping>` element in `<appSettings>` in an application's configuration file, or by setting the `EnableMultiMonitorDisplayClipping` property at app startup.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Framework 4.6.1

Tools

Contract.Invariant or Contract.Requires<TException> do not consider String.IsNullOrEmpty to be pure

Details

For apps that target the .NET Framework 4.6.1, if the invariant contract for [Contract.Invariant](#) or the precondition contract for [Requires](#) calls the [String.IsNullOrEmpty](#) method, the rewriter emits compiler warning CC1036: "Detected call to method 'System.String.IsNullOrEmpty(System.String)' without [Pure] in method." This is a compiler warning rather than a compiler error.

Suggestion

This behavior was addressed in [GitHub Issue #339](#). To eliminate this warning, you can download and compile an updated version of the source code for the Code Contracts tool from [GitHub](#). Download information is found at the bottom of the page.

Name	Value
Scope	Minor
Version	4.6.1
Type	Runtime

Affected APIs

- [Contract.Invariant\(Boolean\)](#)
- [Contract.Requires\(Boolean\)](#)

Windows Presentation Foundation (WPF)

Item-scrolling a flat list with items of different pixel-height

Details

When an [System.Windows.Controls.ItemsControl](#) displays a collection using virtualization (`IsVirtualizing=true`) and item- scrolling (`ScrollUnit=Item`), and when the control scrolls to display an item whose height in pixels differs from its neighbors, the [System.Windows.Controls.VirtualizingStackPanel](#) iterates over all items in the collection. The UI is unresponsive during this iteration. The iteration occurs in other circumstances, even in previous .NET Framework releases. For example, it occurs when pixel-scrolling (`ScrollUnit=Pixel`) upon encountering an item with different pixel height, and when item-scrolling hierarchical data (such as a [System.Windows.Controls.TreeView](#) or an [System.Windows.Controls.ItemsControl](#) with grouping enabled) upon encountering an item with a different number of descendant items than its neighbors. For the case of item-scrolling and different pixel height, the iteration was introduced in .NET Framework 4.6.1 to fix bugs in the layout of hierarchical data. It is not needed if the data is flat (no hierarchy), and .NET Framework 4.6.2 does not do it in that case.

Suggestion

If the iteration occurs in .NET Framework 4.6.1 but not in earlier releases - that is, if the [System.Windows.Controls.ItemsControl](#) is item- scrolling a flat list with items of different pixel height - there are two remedies:

- Install .NET Framework 4.6.2.
- Install hotfix HR 1605 for .NET Framework 4.6.1.

Name	Value
Scope	Minor
Version	4.6.1
Type	Runtime

Affected APIs

- [System.Windows.Controls.VirtualizingStackPanel](#)

ObjectDisposedException thrown by WPF spellchecker

Details

WPF applications occasionally crash during application shutdown with an [System.ObjectDisposedException](#) thrown by the spellchecker. This is fixed in .NET Framework 4.7 WPF by handling the exception gracefully, and thus ensuring that applications are no longer adversely affected. It should be noted that occasional first-chance exceptions would continue to be observed in applications running under a debugger.

Suggestion

Upgrade to .NET Framework 4.7

Name	Value
Scope	Edge
Version	4.6.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF Spell Checking fails in unexpected ways

Details

This includes a number of WPF Spell Checker issues:

- WPF Spell Checker sometimes throws [System.Runtime.InteropServices.COMException](#)
- WPF Spell Checker fails with [UnauthorizedAccessException](#) when applications are launched using 'run as different user'
- WPF Spell Checker incorrectly identifies spelling errors in compound words like 'Hausnummer' in German.

Suggestion

Issue #1 - This has been fixed in .NET Framework 4.6.2 Issue #2 - WPF Spell Checker is no longer supported when applications are launched using 'run as different user'. Starting .NET Framework 4.6.2, applications launched in this manner will no longer crash unexpectedly - instead the Spell Checker will be silently disabled. Issue #3 - This has been fixed in .NET Framework 4.6.2.

Name	Value
Scope	Edge
Version	4.6.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Framework 4.6.2

Data

Attempting a TCP/IP connection to a SQL Server database that resolves to `localhost` fails

Details

In the .NET Framework 4.6 and 4.6.1, attempting a TCP/IP connection to a SQL Server database that resolves to `localhost` fails with the error, "A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: 26 - Error Locating Server/Instance Specified)"

Suggestion

This issue has been addressed and the previous behavior restored in the .NET Framework 4.6.2. To connect to a SQL Server database that resolves to `localhost`, upgrade to the .NET Framework 4.6.2.

Name	Value
Scope	Minor
Version	4.6
Type	Runtime

Affected APIs

Not detectable via API analysis.

Connection pool blocking period for Azure SQL databases is removed

Details

Starting with the .NET Framework 4.6.2, for connection open requests to known Azure SQL databases (*.database.windows.net, *.database.chinacloudapi.cn, *.database.usgovcloudapi.net, *.database.cloudapi.de), the connection pool blocking period is removed, and connection open errors are not cached. Attempts to retry connection open requests will occur almost immediately after transient connection errors. This change allows the connection open attempt to be retried immediately for Azure SQL databases, thereby improving the performance of cloud- enabled apps. For all other connection attempts, the connection pool blocking period continues to be enforced.

In the .NET Framework 4.6.1 and earlier versions, when an app encounters a transient connection failure when connecting to a database, the connection attempt cannot be retried quickly, because the connection pool caches the error and re-throws it for 5 seconds to 1 minute. For more information, see [SQL Server Connection Pooling \(ADO.NET\)](#). This behavior is problematic for connections to Azure SQL databases, which often fail with transient errors that are typically recovered from within a few seconds. The connection pool blocking feature means that the app cannot connect to the database for an extensive period, even though the database is available and the app needs to render within a few seconds.

Suggestion

If this behavior is undesirable, the connection pool blocking period can be configured by setting the [System.Data.SqlClient.SqlConnectionStringBuilder.PoolBlockingPeriod](#)

property introduced in the .NET Framework 4.6.2. The value of the property is a member of the [System.Data.SqlClient.PoolBlockingPeriod](#) enumeration that can take either of three values:

- [AlwaysBlock](#)
- [Auto](#)
- [NeverBlock](#)

The previous behavior can be restored by setting the [System.Data.SqlClient.SqlConnectionStringBuilder.PoolBlockingPeriod](#) property to [AlwaysBlock](#).

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [DbConnection.OpenAsync\(\)](#)
- [SqlConnection.Open\(\)](#)
- [SqlConnection.OpenAsync\(CancellationToken\)](#)

Globalization

Unicode standard version 8.0 categories now supported

Details

In .NET Framework 4.6.2, Unicode data has been upgraded from Unicode Standard version 6.3 to version 8.0. When requesting Unicode character categories in .NET Framework 4.6.2, some results might not match the results in previous .NET Framework versions. This change mostly affects Cherokee syllables and New Tai Lue vowels signs and tone marks.

Suggestion

Review code and remove/change logic that depends on hard-coded Unicode character categories.

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [Char.GetUnicodeCategory\(Char\)](#)
- [CharUnicodeInfo.GetUnicodeCategory\(Char\)](#)
- [CharUnicodeInfo.GetUnicodeCategory\(String, Int32\)](#)

Security

RSACng and DSACng are once again usable in Partial Trust scenarios

Details

CngLightup (used in several higher-level crypto apis, such as [System.Security.Cryptography.Xml.EncryptedXml](#)) and [System.Security.Cryptography.RSACng](#) in some cases rely on full trust. These include P/Invokes without asserting [SecurityPermissionFlag.UnmanagedCode](#) permissions, and code paths where [System.Security.Cryptography.CngKey](#) has permission demands for [SecurityPermissionFlag.UnmanagedCode](#). Starting with the .NET Framework 4.6.2, CngLightup was used to switch to [System.Security.Cryptography.RSACng](#) wherever possible. As a result, partial trust apps that successfully used [System.Security.Cryptography.Xml.EncryptedXml](#) began to fail and throw [SecurityException](#) exceptions. This change adds the required asserts so that all functions using CngLightup have the required permissions.

Suggestion

If this change in the .NET Framework 4.6.2 has negatively impacted your partial trust apps, upgrade to the .NET Framework 4.7.1.

Name	Value
Scope	Edge

Name	Value
Version	4.6.2
Type	Runtime

Affected APIs

- [DSACng\(CngKey\)](#)
- [DSACng.Key](#)
- [DSACng.LegalKeySizes](#)
- [DSACng.CreateSignature\(Byte\[\]\)](#)
- [DSACng.VerifySignature\(Byte\[\], Byte\[\]\)](#)
- [RSACng\(CngKey\)](#)
- [RSACng.Key](#)
- [RSACng.Decrypt\(Byte\[\], RSAEncryptionPadding\)](#)
- [RSACng.SignHash\(Byte\[\], HashAlgorithmName, RSASignaturePadding\)](#)

RSACng.VerifyHash now returns False for any verification failure

Details

Starting with the .NET Framework 4.6.2, this method returns **False** if the signature itself is badly formatted. It now returns false for any verification failure. In the .NET Framework 4.6 and 4.6.1, the method throws a

[System.Security.Cryptography.CryptographicException](#) if the signature itself is badly formatted.

Suggestion

Any code whose execution depends on handling the [System.Security.Cryptography.CryptographicException](#) should instead execute if validation fails and the method returns **False**.

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [RSACng.VerifyHash\(Byte\[\], Byte\[\], HashAlgorithmName, RSASignaturePadding\)](#)

SignedXml and EncryptedXml Breaking Changes

Details

In .NET Framework 4.6.2, security fixes in [System.Security.Cryptography.Xml.SignedXml](#) and [System.Security.Cryptography.Xml.EncryptedXml](#) lead to different run-time behaviors. For example:

- If a document has multiple elements with the same `id` attribute and a signature targets one of those elements as the root of the signature, the document will now be considered invalid.
- Documents using non-canonical XPath transform algorithms in references are now considered invalid.
- Documents using non-canonical XSLT transform algorithms in references are now consider invalid.
- Any program making use of external resource detached signatures will be unable to do so.

Suggestion

Developers might want to review the usage of [XmlDsigXsltTransform](#) and [XmlDsigXsltTransform](#), as well as types derived from [Transform](#) since a document receiver may not be able to process it.

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [System.Security.Cryptography.Xml.Transform](#)
- [System.Security.Cryptography.Xml.XmlDsigXPathTransform](#)
- [System.Security.Cryptography.Xml.XmlDsigXsltTransform](#)

Windows Communication Foundation (WCF)

Remove Ssl3 from the WCF TransportDefaults

Details

When using NetTcp with transport security and a credential type of certificate, the SSL 3 protocol is no longer a default protocol used for negotiating a secure connection. In most cases there should be no impact to existing apps as TLS 1.0 has always been included in the protocol list for NetTcp. All existing clients should be able to negotiate a connection using at least TLS1.0.

Suggestion

If Ssl3 is required, use one of the following configuration mechanisms to add Ssl3 to the list of negotiated protocols.

- [SslProtocols](#)
- [SslProtocols](#)
- [`<transport> of <netTcpBinding>`](#)
- [`<sslStreamSecurity>`](#)

Name	Value
Scope	Edge
Version	4.6.2
Type	Runtime

Affected APIs

- [SslStreamSecurityBindingElement.SslProtocols](#)
- [TcpTransportSecurity.SslProtocols](#)

Windows Presentation Foundation (WPF)

Changing the IsEnabled property of the parent of a TextBlock control affects any child controls

Details

Starting with the .NET Framework 4.6.2, changing the [System.Windows.UIElement.IsEnabled](#) property of the parent of a [System.Windows.Controls.TextBlock](#) control affects any child controls (such as hyperlinks and buttons) of the [System.Windows.Controls.TextBlock](#) control. In the .NET Framework 4.6.1 and earlier versions, controls inside a [System.Windows.Controls.TextBlock](#) did not always reflect the state of the [System.Windows.UIElement.IsEnabled](#) property of the [System.Windows.Controls.TextBlock](#) parent.

Suggestion

None. This change conforms to the expected behavior for controls inside a [System.Windows.Controls.TextBlock](#) control.

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [UIElement.IsEnabled](#)

CoercelsSelectionBoxHighlighted

Details

Certain sequences of actions involving a [System.Windows.Controls.ComboBox](#) and its data source can result in a [System.NullReferenceException](#).

Suggestion

If possible, upgrade to .NET Framework 4.6.2.

Name	Value
Scope	Minor

Name	Value
Version	4.6
Type	Runtime

Affected APIs

- [ComboBox.IsSelectionBoxHighlighted](#)

DataGridCellsPanel.BringIndexIntoView throws ArgumentException

Details

[ScrollIntoView\(Object\)](#) will work asynchronously when column virtualization is enabled but the column widths have not yet been determined. If columns are removed before the asynchronous work happens, an [System.ArgumentOutOfRangeException](#) can occur.

Suggestion

Any one of the following:

- Upgrade to .NET Framework 4.7.
- Install the latest servicing patch for .NET Framework 4.6.2.
- Avoid removing columns until the asynchronous response to [ScrollIntoView\(Object\)](#) has completed.

Name	Value
Scope	Edge
Version	4.6.2
Type	Runtime

Affected APIs

- [DataGrid.ScrollIntoView\(Object\)](#)
- [DataGrid.ScrollIntoView\(Object, DataGridColumn\)](#)

Horizontal scrolling and virtualization

Details

This change applies to an [System.Windows.Controls.ItemsControl](#) that does its own virtualization in the direction orthogonal to the main scrolling direction (the chief example is [System.Windows.Controls.DataGrid](#) with `EnableColumnVirtualization=True`). The outcome of certain horizontal scrolling operations has been changed to produce results that are more intuitive and more analogous to the results of comparable vertical operations.

The operations include "Scroll Here" and "Right Edge", to use the names from the menu obtained by right-clicking a horizontal scrollbar. Both of these compute a candidate offset and call [SetHorizontalOffset\(Double\)](#).

After scrolling to the new offset, the notion of "here" or "right edge" may have changed because newly de-virtualized content has changed the value of [System.Windows.Controls.Primitives.IScrollInfo.ExtentWidth](#).

Prior to .NET Framework 4.6.2, the scroll operation simply uses the candidate offset, even though it may not be "here" or at the "right edge" any more. This results in effects like "bouncing" the scroll thumb, best illustrated by example. Suppose a [System.Windows.Controls.DataGrid](#) has `ExtentWidth=1000` and `Width=200`. A scroll to "Right Edge" uses candidate offset $1000 - 200 = 800$. While scrolling to that offset, new columns are de- virtualized; let's suppose they are very wide, so that the [System.Windows.Controls.Primitives.IScrollInfo.ExtentWidth](#) changes to 2000. The scroll ends with `HorizontalOffset=800`, and the thumb "bounces" back to near the middle of the scrollbar - precisely at $800/2000 = 40\%$.

The change is to recompute a new candidate offset when this situation occurs, and try again. (This is how vertical scrolling works already.)

The change produces a more predictable and intuitive experience for the end user, but it could also affect any app that depends on the exact value of [System.Windows.Controls.Primitives.IScrollInfo.HorizontalOffset](#) after a horizontal scroll, whether invoked by the end user or by an explicit call to [SetHorizontalOffset\(Double\)](#).

Suggestion

An app that uses a predicted value for [System.Windows.Controls.Primitives.IScrollInfo.HorizontalOffset](#) should be changed to fetch the actual value (and the value of [System.Windows.Controls.Primitives.IScrollInfo.ExtentWidth](#)) after any horizontal scroll that could change [System.Windows.Controls.Primitives.IScrollInfo.ExtentWidth](#) due to de-virtualization.

Name	Value
Scope	Minor
Version	4.6.2
Type	Runtime

Affected APIs

- [System.Windows.Controls.Primitives.IScrollInfo](#)

Items.Clear does not remove duplicates from SelectedItems

Details

Suppose a Selector (with multiple selection enabled) has duplicates in its [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) collection - the same item appears more than once. Removing those items from the data source (e.g. by calling Items.Clear) fails to remove them from [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#); only the first instance is removed. Furthermore, subsequent use of [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) (e.g. SelectedItems.Clear()) can encounter problems such as [System.ArgumentException](#), because [System.Windows.Controls.Primitives.MultiSelector.SelectedItems](#) contains items that are no longer in the data source.

Suggestion

Upgrade if possible to .NET Framework 4.6.2.

Name	Value
Scope	Minor
Version	4.5
Type	Runtime

Affected APIs

- MultiSelector.SelectedItems

Item-scrolling a flat list with items of different pixel-height

Details

When an [System.Windows.Controls.ItemsControl](#) displays a collection using virtualization (`IsVirtualizing=true`) and item- scrolling (`ScrollUnit=Item`), and when the control scrolls to display an item whose height in pixels differs from its neighbors, the [System.Windows.Controls.VirtualizingStackPanel](#) iterates over all items in the collection. The UI is unresponsive during this iteration. The iteration occurs in other circumstances, even in previous .NET Framework releases. For example, it occurs when pixel-scrolling (`ScrollUnit=Pixel`) upon encountering an item with different pixel height, and when item-scrolling hierarchical data (such as a [System.Windows.Controls.TreeView](#) or an [System.Windows.Controls.ItemsControl](#) with grouping enabled) upon encountering an item with a different number of descendant items than its neighbors. For the case of item-scrolling and different pixel height, the iteration was introduced in .NET Framework 4.6.1 to fix bugs in the layout of hierarchical data. It is not needed if the data is flat (no hierarchy), and .NET Framework 4.6.2 does not do it in that case.

Suggestion

If the iteration occurs in .NET Framework 4.6.1 but not in earlier releases - that is, if the [System.Windows.Controls.ItemsControl](#) is item- scrolling a flat list with items of different pixel height - there are two remedies:

- Install .NET Framework 4.6.2.
- Install hotfix HR 1605 for .NET Framework 4.6.1.

Name	Value
Scope	Minor
Version	4.6.1
Type	Runtime

Affected APIs

- [System.Windows.Controls.VirtualizingStackPanel](#)

RibbonGroup background is set to transparent in localized builds

Details

[System.Windows.Controls.Ribbon.RibbonGroup](#) background on localized builds was always painted with Transparent brush, resulting in poor UI experience. This is fixed in .NET Framework 4.7 WPF fix by updating the localized resources for [System.Windows.Controls.Ribbon.RibbonGroup](#), which in turn ensures that the correct brush is selected.

Suggestion

Upgrade to .NET Framework 4.7

Name	Value
Scope	Edge
Version	4.6.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF Spell Checking fails in unexpected ways

Details

This includes a number of WPF Spell Checker issues:

- WPF Spell Checker sometimes throws [System.Runtime.InteropServices.COMException](#)
- WPF Spell Checker fails with [UnauthorizedAccessException](#) when applications are launched using 'run as different user'
- WPF Spell Checker incorrectly identifies spelling errors in compound words like 'Hausnummer' in German.

Suggestion

Issue #1 - This has been fixed in .NET Framework 4.6.2 Issue #2 - WPF Spell Checker is no longer supported when applications are launched using 'run as different user'. Starting .NET Framework 4.6.2, applications launched in this manner will no longer crash unexpectedly - instead the Spell Checker will be silently disabled. Issue #3 - This has been fixed in .NET Framework 4.6.2.

Name	Value
Scope	Edge
Version	4.6.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Runtime changes for migration to .NET Framework 4.7.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.7](#), [4.7.1](#), and [4.7.2](#).

.NET Framework 4.7

JIT

Incorrect code generation when passing and comparing UInt16 values

Details

Because of changes introduced in the .NET Framework 4.7, in some cases the code generated by the JIT compiler in applications running on the .NET Framework 4.7 incorrectly compares two `T:System.UInt16` values. For more information, see [Issue #11508: Silent bad codegen when passing and comparing ushort args](#) on GitHub.com.

Suggestion

If you encounter issues in the comparison of 16-bit unsigned values in the .NET Framework 4.7, upgrade to the .NET Framework 4.7.1.

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Crash in Selector when removing an item from a custom INCC collection

Details

An `T:System.InvalidOperationException` can occur in the following scenario:

- The `ItemsSource` for a `T:System.Windows.Controls.Primitives.Selector` is a collection with a custom implementation of `T:System.Collections.Specialized.INotifyCollectionChanged`.
- The selected item is removed from the collection.
- The `T:System.Collections.Specialized.NotifyCollectionChangedEventArgs` has `P:System.Collections.Specialized.NotifyCollectionChangedEventArgs.OldStartingIndex` = -1 (indicating an unknown position).

The exception's call stack begins at

```
System.Windows.Threading.Dispatcher.VerifyAccess() at  
System.Windows.DependencyObject.GetValue(DependencyProperty dp) at  
System.Windows.Controls.Primitives.Selector.GetIsSelected(DependencyObject  
element). This exception can occur in .NET Framework 4.5 if the application has more  
than one Dispatcher thread. In .NET Framework 4.7 the exception can also occur in  
applications with a single Dispatcher thread.
```

The issue is fixed in .NET Framework 4.7.1.

Suggestion

Upgrade to .NET Framework 4.7.1.

Name	Value
Scope	Minor
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

DataGridCellsPanel.BringIndexIntoView throws ArgumentException

Details

[ScrollIntoView\(Object\)](#) will work asynchronously when column virtualization is enabled but the column widths have not yet been determined. If columns are removed before the asynchronous work happens, an [System.ArgumentOutOfRangeException](#) can occur.

Suggestion

Any one of the following:

- Upgrade to .NET Framework 4.7.
- Install the latest servicing patch for .NET Framework 4.6.2.
- Avoid removing columns until the asynchronous response to [ScrollIntoView\(Object\)](#) has completed.

Name	Value
Scope	Edge
Version	4.6.2
Type	Runtime

Affected APIs

- [DataGrid.ScrollIntoView\(Object\)](#)
- [DataGrid.ScrollIntoView\(Object, DataGridColumn\)](#)

ObjectDisposedException thrown by WPF spellchecker

Details

WPF applications occasionally crash during application shutdown with an [System.ObjectDisposedException](#) thrown by the spellchecker. This is fixed in .NET Framework 4.7 WPF by handling the exception gracefully, and thus ensuring that applications are no longer adversely affected. It should be noted that occasional first-

chance exceptions would continue to be observed in applications running under a debugger.

Suggestion

Upgrade to .NET Framework 4.7

Name	Value
Scope	Edge
Version	4.6.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Resizing a Grid can cause the application to become unresponsive

Details

An infinite loop can occur during layout of a `T:System.Windows.Controls.Grid` under the following circumstances:

- Row definitions contain two *-rows, both declaring a MinHeight and a MaxHeight.
- Content of the *-rows doesn't exceed the corresponding MaxHeight.
- The Grid's available height is exceeded by the first MinHeight (plus any other fixed or Auto rows).
- The app targets .NET Framework 4.7, or opts in to the 4.7 allocation algorithm by setting

```
Switch.System.Windows.Controls.Grid.StarDefinitionsCanExceedAvailableSpace=false
```

The loop would also happen with more than two rows, or in the analogous case for columns. The issue is fixed in .NET Framework 4.7.1.

Suggestion

Upgrade to .NET Framework 4.7.1. Alternatively, if you don't need the 4.7 allocation algorithm you can use the following configuration setting:

XML
<pre><runtime> <AppContextSwitchOverrides value="Switch.System.Windows.Controls.Grid.StarDefinitionsCanExceedAvailable Space=true" /> </runtime></pre>

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

RibbonGroup background is set to transparent in localized builds

Details

[System.Windows.Controls.Ribbon.RibbonGroup](#) background on localized builds was always painted with Transparent brush, resulting in poor UI experience. This is fixed in .NET Framework 4.7 WPF fix by updating the localized resources for [System.Windows.Controls.Ribbon.RibbonGroup](#), which in turn ensures that the correct brush is selected.

Suggestion

Upgrade to .NET Framework 4.7

Name	Value
Scope	Edge
Version	4.6.2

Name	Value
Type	Runtime

Affected APIs

Not detectable via API analysis.

WPF Printing Stack Update

Details

WPF's Printing APIs using [System.Printing.PrintQueue](#) now call Window's Print Document Package API in favor of the now deprecated XPS Print API. The change was made with serviceability in mind; neither users nor developers should see any changes in behavior or API usage. The new printing stack is enabled by default when running in Windows 10 Creators Update. The old printing stack will still continue to work just as before in older Windows versions.

Suggestion

To use the old stack in Windows 10 Creators Update, set the [UseXpsOMPrinting](#) REG_DWORD value of the [HKEY_CURRENT_USER\Software\Microsoft\.NETFramework\Windows Presentation Foundation\Printing](#) registry key to [1](#).

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Workflow Foundation (WF)

Workflow now throws original exception instead of NullReferenceException in some cases

Details

In the .NET Framework 4.6.2 and earlier versions, when the Execute method of a workflow activity throws an exception with a `null` value for the [Message](#) property, the System.Activities Workflow runtime throws a [System.NullReferenceException](#), masking the original exception. In the .NET Framework 4.7, the previously masked exception is thrown.

Suggestion

If your code relies on handling the [System.NullReferenceException](#), change it to catch the exceptions that could be thrown from your custom activities.

Name	Value
Scope	Minor
Version	4.7
Type	Runtime

Affected APIs

- [CodeActivity.Execute\(CodeActivityContext\)](#)
- [AsyncCodeActivity.BeginExecute\(AsyncCodeActivityContext, AsyncCallback, Object\)](#)
- [AsyncCodeActivity<TResult>.BeginExecute\(AsyncCodeActivityContext, AsyncCallback, Object\)](#)
- [WorkflowInvoker.Invoke\(\)](#)

Workflow SQL persistence adds primary key clusters and disallows null values in some columns

Details

Starting with the .NET Framework 4.7, the tables created for the SQL Workflow Instance Store (SWIS) by the SqlWorkflowInstanceStoreSchema.sql script use clustered primary keys. Because of this, identities do not support `null` values. The operation of SWIS is

not impacted by this change. The updates were made to support SQL Server Transactional Replication.

Suggestion

The SQL file SqlWorkflowInstanceStoreSchemaUpgrade.sql must be applied to existing installations in order to experience this change. New database installations will automatically have the change.

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Framework 4.7.1

JIT

Incorrect code generation when passing and comparing UInt16 values

Details

Because of changes introduced in the .NET Framework 4.7, in some cases the code generated by the JIT compiler in applications running on the .NET Framework 4.7 incorrectly compares two `T:System.UInt16` values. For more information, see [Issue #11508: Silent bad codegen when passing and comparing ushort args](#) on GitHub.com.

Suggestion

If you encounter issues in the comparison of 16-bit unsigned values in the .NET Framework 4.7, upgrade to the .NET Framework 4.7.1.

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

Security

RSACng and DSACng are once again usable in Partial Trust scenarios

Details

CngLightup (used in several higher-level crypto apis, such as [System.Security.Cryptography.Xml.EncryptedXml](#)) and [System.Security.Cryptography.RSACng](#) in some cases rely on full trust. These include P/Invokes without asserting [SecurityPermissionFlag.UnmanagedCode](#) permissions, and code paths where [System.Security.Cryptography.CngKey](#) has permission demands for [SecurityPermissionFlag.UnmanagedCode](#). Starting with the .NET Framework 4.6.2, CngLightup was used to switch to [System.Security.Cryptography.RSACng](#) wherever possible. As a result, partial trust apps that successfully used [System.Security.Cryptography.Xml.EncryptedXml](#) began to fail and throw [SecurityException](#) exceptions. This change adds the required asserts so that all functions using CngLightup have the required permissions.

Suggestion

If this change in the .NET Framework 4.6.2 has negatively impacted your partial trust apps, upgrade to the .NET Framework 4.7.1.

Name	Value
Scope	Edge
Version	4.6.2

Name	Value
Type	Runtime

Affected APIs

- [DSACng\(CngKey\)](#)
- [DSACng.Key](#)
- [DSACng.LegalKeySizes](#)
- [DSACng.CreateSignature\(Byte\[\]\)](#)
- [DSACng.VerifySignature\(Byte\[\], Byte\[\]\)](#)
- [RSACng\(CngKey\)](#)
- [RSACng.Key](#)
- [RSACng.Decrypt\(Byte\[\], RSAEncryptionPadding\)](#)
- [RSACng.SignHash\(Byte\[\], HashAlgorithmName, RSASignaturePadding\)](#)

Windows Communication Foundation (WCF)

WCF AddressHeaderCollection now throws an ArgumentException if an addressHeader element is null

Details

Starting with the .NET Framework 4.7.1, the [AddressHeaderCollection\(IEnumerable<AddressHeader>\)](#) constructor throws an [ArgumentException](#) if one of the elements is `null`. In the .NET Framework 4.7 and earlier versions, no exception is thrown.

Suggestion

If you encounter compatibility issues with this change on the .NET Framework 4.7.1 or a later version, you can opt-out of it by adding the following line to the `<runtime>` section of the app.config file:

XML

```

<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.ServiceModel.DisableAddressHeaderCollectionValidation=t
rue" />

```

```
</runtime>
</configuration>
```

Name	Value
Scope	Minor
Version	4.7.1
Type	Runtime

Affected APIs

- [AddressHeaderCollection\(IEnumerable<AddressHeader>\)](#)

WCF MsmqSecureHashAlgorithm default value is now SHA256

Details

Starting with the .NET Framework 4.7.1, the default message signing algorithm in WCF for Msmq messages is SHA256. In the .NET Framework 4.7 and earlier versions, the default message signing algorithm is SHA1.

Suggestion

If you run into compatibility issues with this change on the .NET Framework 4.7.1 or later, you can opt-out the change by adding the following line to the `<runtime>` section of your app.config file:

XML

```
<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.ServiceModel.UseSha1InMsmqEncryptionAlgorithm=true" />
  </runtime>
</configuration>
```

Name	Value
Scope	Minor

Name	Value
Version	4.7.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

WCF PipeConnection.GetHashAlgorithm now uses SHA256

Details

Starting with the .NET Framework 4.7.1, Windows Communication Foundation uses a SHA256 hash to generate random names for named pipes. In the .NET Framework 4.7 and earlier versions, it used a SHA1 hash.

Suggestion

If you run into compatibility issue with this change on the .NET Framework 4.7.1 or later, you can opt-out it by adding the following line to the `<runtime>` section of your `app.config` file:

XML

```

<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.ServiceModel.UseSha1InPipeConnectionGetHashAlgorithm=true" />
  </runtime>
</configuration>

```

Name	Value
Scope	Minor
Version	4.7.1
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Chained Popups with StaysOpen=False

Details

A Popup with StaysOpen=False is supposed to close when you click outside the Popup. When two or more such Popups are chained (i.e. one contains another), there were many problems, including:

- Open two levels, click outside P2 but inside P1. Nothing happens.
- Open two levels, click outside P1. Both popups close.
- Open and close two levels. Then try to open P2 again. Nothing happens.
- Try to open three levels. You can't. (Either nothing happens or the first two levels close, depending on where you click.)

These cases (and other variants) now work as expected.

Name	Value
Scope	Edge
Version	4.7.1
Type	Runtime

Affected APIs

- [Popup.StaysOpen](#)

Crash in Selector when removing an item from a custom INCC collection

Details

An `T:System.InvalidOperationException` can occur in the following scenario:

- The ItemsSource for a `T:System.Windows.Controls.Primitives.Selector` is a collection with a custom implementation of `T:System.Collections.Specialized.INotifyCollectionChanged`.
- The selected item is removed from the collection.
- The `T:System.Collections.Specialized.NotifyCollectionChangedEventArgs` has `P:System.Collections.Specialized.NotifyCollectionChangedEventArgs.OldStartingIndex` = -1 (indicating an unknown position).

The exception's call stack begins at

```
System.Windows.Threading.Dispatcher.VerifyAccess() at
System.Windows.DependencyObject.GetValue(DependencyProperty dp) at
System.Windows.Controls.Primitives.Selector.GetIsSelected(DependencyObject
element). This exception can occur in .NET Framework 4.5 if the application has more
than one Dispatcher thread. In .NET Framework 4.7 the exception can also occur in
applications with a single Dispatcher thread.
```

The issue is fixed in .NET Framework 4.7.1.

Suggestion

Upgrade to .NET Framework 4.7.1.

Name	Value
Scope	Minor
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

Resizing a Grid can cause the application to become unresponsive

Details

An infinite loop can occur during layout of a `T:System.Windows.Controls.Grid` under the following circumstances:

- Row definitions contain two *-rows, both declaring a MinHeight and a MaxHeight.
- Content of the *-rows doesn't exceed the corresponding MaxHeight.
- The Grid's available height is exceeded by the first MinHeight (plus any other fixed or Auto rows).
- The app targets .NET Framework 4.7, or opts in to the 4.7 allocation algorithm by setting

```
Switch.System.Windows.Controls.Grid.StarDefinitionsCanExceedAvailableSpace=fal
se.
```

The loop would also happen with more than two rows, or in the analogous case for columns. The issue is fixed in .NET Framework 4.7.1.

Suggestion

Upgrade to .NET Framework 4.7.1. Alternatively, if you don't need the 4.7 allocation algorithm you can use the following configuration setting:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Windows.Controls.Grid.StarDefinitionsCanExceedAvailable
    Space=true" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.7
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Framework 4.7.2

Core

Allow Unicode in URIs that resemble UNC shares

Details

In [System.Uri](#), constructing a file URI containing both a UNC share name and Unicode characters will no longer result in a URI with invalid internal state. The behavior will change only when all of the following are true:

- The URI has the scheme `file:` and is followed by four or more slashes.
- The host name begins with an underscore or other non-reserved symbol.
- The URI contains Unicode characters.

Suggestion

Applications working with URIs consistently containing Unicode could have conceivably used this behavior to disallow references to UNC shares. Those applications should use [IsUnc](#) instead.

Name	Value
Scope	Edge
Version	4.7.2
Type	Runtime

Affected APIs

- [System.Uri](#)

Support special relative URI notation when Unicode is present

Details

[Uri](#) will no longer throw a [NullReferenceException](#) when calling [TryCreate](#) on certain relative URIs containing Unicode. The simplest reproduction of the [NullReferenceException](#) is below, with the two statements being equivalent:

C#

```
bool success = Uri.TryCreate("http:%C3%A8", UriKind.RelativeOrAbsolute, out Uri href);
bool success = Uri.TryCreate("http:è", UriKind.RelativeOrAbsolute, out Uri href);
```

To reproduce the [NullReferenceException](#), the following items must be true:

- The URI must be specified as relative by prepending it with 'http:' and not following it with '//'.
- The URI must contain percent-encoded Unicode or unreserved symbols.

Suggestion

Users depending on this behavior to disallow relative URIs should instead specify [UriKind.Absolute](#) when creating a URI.

Name	Value
Scope	Edge
Version	4.7.2
Type	Runtime

Affected APIs

- [Uri.TryCreate\(Uri, Uri, Uri\)](#)
- [Uri.TryCreate\(String, UriKind, Uri\)](#)
- [Uri.TryCreate\(Uri, String, Uri\)](#)

Runtime

Improved WCF chain trust certificate validation for Net.Tcp certificate authentication

Details

.NET Framework 4.7.2 improves chain trust certificate validation when using certificate authentication with transport security with WCF. With this improvement, client certificates that are used to authenticate to a server must be configured for client authentication. Similarly server certificates that are for the authenticating a server must be configured for server authentication. With this change, if the root certificate is disabled, the certificate chain validation fails. The same change was also made to .NET Framework 3.5 and later versions via Windows security roll-up. You can find more information [here ↗](#). This change is on by default and can be turned off by a configuration setting.

Suggestion

- Validate if your server and client certification has the required EKU OID. If not, update your certification.
- Validate if your root certificate is invalid. If so, update the root certificate.
- If you can't update the certificate, you can work around the breaking change temporarily with the following configuration setting. However, opting out of the change will leave your system vulnerable to the security issue.

XML

```
<appSettings>
  <add key="wcf:useLegacyCertificateUsagePolicy" value="true" />
</appSettings>
```

Name	Value
Scope	Minor
Version	4.7.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Web Applications

"dataAnnotations:dataTypeAttribute:disableRegEx" app setting is on by default in .NET Framework 4.7.2

Details

In .NET Framework 4.6.1, an app setting

(`"dataAnnotations:dataTypeAttribute:disableRegEx"`) was introduced that allows users to disable the use of regular expressions in data type attributes (such as [System.ComponentModel.DataAnnotations.EmailAddressAttribute](#), [System.ComponentModel.DataAnnotations.UrlAttribute](#), and [System.ComponentModel.DataAnnotations.PhoneAttribute](#)). This helps to reduce

security vulnerability such as avoiding the possibility of a Denial of Service attack using specific regular expressions.

In .NET Framework 4.6.1, this app setting to disable RegEx usage was set to `false` by default. Starting with .NET Framework 4.7.2, this config switch is set to `true` by default to further reduce secure vulnerability for web applications that target .NET Framework 4.7.2 and above.

Suggestion

If you find that regular expressions in your web application do not work after upgrading to .NET Framework 4.7.2, you can update the value of the `"dataAnnotations:dataTypeAttribute:disableRegEx"` setting to `false` to revert to the previous behavior.

XML

```
<configuration>
<appSettings>
...
<add key="dataAnnotations:dataTypeAttribute:disableRegEx" value="false"/>
...
</appSettings>
</configuration>
```

Name	Value
Scope	Minor
Version	4.7.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Keytips behavior improved in WPF

Details

Keytips behavior has been modified to bring parity with behavior on Microsoft Word and Windows Explorer. By checking whether keytip state is enabled or not in the case of a [SystemKey](#) (in particular, [Key](#) or [F11](#)) being pressed, WPF handles keytip keys appropriately. Keytips now dismiss a menu even when it is opened by mouse.

Suggestion

N/A

Name	Value
Scope	Edge
Version	4.7.2
Type	Runtime

Affected APIs

Not detectable via API analysis.

Runtime changes for migration to .NET Framework 4.8.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.8](#) and [4.8.1](#).

.NET Framework 4.8

ASP.NET

ASP.NET Fix handling of InputAttributes and LabelAttributes for WebForms CheckBox control

Details

For applications that target .NET Framework 4.7.2 and earlier versions, [CheckBox.InputAttributes](#) and [CheckBox.LabelAttributes](#) that are programmatically added to a WebForms [CheckBox](#) control are lost after postback. For applications that target .NET Framework 4.8 or later versions, they are preserved after postback.

Suggestion

For the correct behavior for restoring attributes on postback, set the `targetFrameworkVersion` to 4.8 or higher. For example:

XML

```
<configuration>
<system.web>
<httpRuntime targetFramework="4.8"/>
</system.web>
</configuration>
```

Setting it lower, or not at all, preserves the old incorrect behavior.

Name	Value
Scope	Unknown

Name	Value
Version	4.8
Type	Runtime

Affected APIs

- [System.Web.UI.WebControls.CheckBox](#)

ASP.NET Incorrect multipart handling may result in lost form data.

Details

In applications that target .NET Framework 4.7.2 and earlier versions, ASP.NET might incorrectly parse multipart boundary values, resulting in form data being unavailable during request execution. Applications that target .NET Framework 4.8 or later versions correctly parse multipart data, so form values are available during request execution.

Suggestion

Starting with applications running on .NET Framework 4.8, when targeting .NET Framework 4.8 or later by using the `targetFrameworkVersion` element, the default behavior changes to strip delimiters. When targeting previous framework versions or not using `targetFrameworkVersion`, trailing delimiters for some values are still returned.

This behavior can also be explicitly controlled with an `appSetting`:

XML

<pre><configuration> <appSettings> ... <add key="aspnet:UseLegacyMultiValueHeaderHandling" value="true"/> ... </appSettings> </configuration></pre>

Name	Value
Scope	Unknown

Name	Value
Version	4.8
Type	Runtime

Affected APIs

- [HttpRequest.Form](#)
- [HttpRequest.Files](#)
- [HttpRequest.ContentEncoding](#)

ASP.NET ValidationContext.MemberName is not NULL when using custom DataAnnotations.ValidationAttribute

Details

In .NET Framework 4.7.2 and earlier versions, when using a custom [System.ComponentModel.DataAnnotations.ValidationAttribute](#), the [ValidationContext.MemberName](#) property returns `null`. In .NET Framework 4.8 version prior to the October 2019 update, it returns the member name. Starting with [.NET Framework October 2019 Preview of Quality Rollup](#) for .NET Framework 4.8, it returns `null` by default, but you can opt in to return the member name instead.

Suggestion

Add the following setting to your *web.config* file for the property to return the member name in [.NET Framework October 2019 Preview of Quality Rollup](#) for .NET Framework 4.8 and later versions:

XML

```
<configuration>
<appSettings>
...
<add key="aspnet:GetValidationMemberName" value="true"/>
...
</appSettings>
</configuration>
```

In .NET Framework 4.8 version prior to the October 2019 update, adding this to your *web.config* file restores the previous behavior and the property returns `null`.

Name	Value
Scope	Unknown
Version	4.8
Type	Runtime

Affected APIs

- [ValidationContext.MemberName](#)

Core

.NET COM successfully marshals ByRef SafeArray parameters on events

Details

In .NET Framework 4.7.2 and earlier versions, a ByRef [SafeArray](#) parameter on a COM event would fail to marshal back to native code. With this change, the [SafeArray](#) is now marshalled successfully.

- [x] Quirked

Suggestion

If properly marshalling ByRef SafeArray parameters on COM Events breaks execution, you can disable this code by adding the following configuration switch to your application config:

XML

```
<appSettings>
  <add
    key="Switch.System.Runtime.InteropServices.DoNotMarshalOutByrefSafeArrayOnInvoke"
    value="true" />
</appSettings>
```

Name	Value
Scope	Minor

Name	Value
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Interop will now QueryInterface for IAgileObject (a WinRT interface)

Details

When using a WinRT event with a .NET delegate, Windows will QI for IAgileObject starting with .NET Framework 4.8. In previous versions of .NET Framework, the runtime would fail that QI, and the event could not be subscribed.

- [x] Quirked

Suggestion

If enabling the QI for IAgileObject breaks execution, you can disable this code by setting the following configuration.

Method 1: Environment variable

Set the following environment variable: `COMPLUS_DisableCCWSupportIAgileObject=1`

This method affects any environment that inherits this environment variable. This might be just a single console session, or it might affect the entire machine if you set the environment variable globally. The environment variable name is not case-sensitive.

Method 2: Registry

Using Registry Editor (regedit.exe), find either of the following subkeys:

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework`
- `HKEY_CURRENT_USER\SOFTWARE\Microsoft.NETFramework`

Then add the following entry:

Name: DisableCCWSupportIAgileObject Type: DWORD (32-bit) value (also called REG_DWORD) Data: 1

You can use the Windows REG.EXE tool to add this value from a command line or scripting environment. For example:

Console

```
reg add HKLM\SOFTWARE\Microsoft.NETFramework /v  
DisableCCWSupportIAgileObject /t REG_DWORD /d 1
```

In this case, `HKLM` is used instead of `HKEY_LOCAL_MACHINE`. Use `reg add /?` to see help on this syntax. The registry value name is not case-sensitive.

Name	Value
Scope	Edge
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Communication Foundation (WCF)

svcTraceViewer ComboBox high contrast change

Details

In the [Microsoft Service Trace Viewer tool](#), ComboBox controls were not displayed in the correct color in certain high contrast themes. The issue was fixed in .NET Framework 4.7.2. However, due to .NET Framework SDK backward compatibility requirements, the fix was not visible to customers by default. .NET 4.8 surfaces this change by adding the following [AppContext configuration switches](#) to the svcTraceViewer.exe.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi  
lityFeatures.2=false" />
```

Suggestion

If you don't want to have the high contrast behavior change, you can disable it by removing the following section from the svcTraceViewer.exe.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi  
lityFeatures.2=false" />
```

Name	Value
Scope	Edge
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

Windows Presentation Foundation (WPF)

Data Binding improvement for KeyedCollection

Details

Fixed [Binding](#) incorrect use of IList indexer when the source object declares a custom indexer with the same signature (for example, KeyedCollection<int,TItem>).

Suggestion

In order for an application that targets an older version to benefit from this change, it must run on the .NET Framework 4.8 or later, and it must opt in to the change by adding

the following `AppContext switch` to the `<runtimes>` section of the app config file and setting it to `false`:

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
</startup>
<runtimes>
<!-- AppContextSwitchOverrides value attribute is in the form of
'key1=true/false;key2=true/false -->
<AppContextSwitchOverrides
value="Switch.System.Windows.Data.Binding_IListIndexerHidesCustomIndexer=false" />
</runtimes>
</configuration>
```

Name	Value
Scope	Major
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

Fixed a issue when ListBox stops responding if it contains duplicate value-types

Details

Fixed a problem where a virtualizing `ItemsControl` can stop responding during scrolling when its `Items` collection contains duplicate value-typed objects.

Name	Value
Scope	Major
Version	4.8

Name	Value
Type	Runtime

Affected APIs

Not detectable via API analysis.

Improvements to Grid star-rows space allocating algorithm

Details

Fixed a bug in the [algorithm for allocating sizes to](#)) in a `Grid` introduced in .NET Framework 4.7. In some cases, such as a Grid with `Height="Auto"` containing empty rows, rows were arranged at the wrong position, possibly outside the Grid altogether.

Suggestion

In order for the application to benefit from these changes, it must run on the .NET Framework 4.8 or later.

Name	Value
Scope	Major
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

Keyboard navigation improvement in ListBox with Hyperlinks

Details

Fixed incorrect result of pressing an arrow key when the focus is on a hyperlink within an item that is not the selected item of the parent [ItemsControl](#).

Name	Value
Scope	Major
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

Performance improvement in Automation tree for grouping ItemsControls

Details

Improved the performance of rebuilding the automation tree of an [ItemsControl](#), such as a ListBox or DataGridView, in which grouping is enabled.

Name	Value
Scope	Major
Version	4.8
Type	Runtime

Affected APIs

Not detectable via API analysis.

.NET Framework 4.8.1

No app compatibility issues were introduced in .NET Framework 4.8.1.

Retargeting changes for migration to .NET Framework 4.5.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.5](#), [4.5.1](#), and [4.5.2](#).

.NET Framework 4.5

ASP.NET

MachineKey.Encode and MachineKey.Decode methods are now obsolete

Details

These methods are now obsolete. Compilation of code that calls these methods produces a compiler warning.

Suggestion

The recommended alternatives are [Protect\(Byte\[\], String\[\]\)](#) and [Unprotect\(Byte\[\], String\[\]\)](#). Alternatively, the build warnings can be suppressed, or they can be avoided by using an older compiler. The APIs are still supported.

[+] Expand table

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

Affected APIs

- [MachineKey.Encode\(Byte\[\], MachineKeyProtection\)](#)

- `MachineKey.Decode(String, MachineKeyProtection)`

Multi-line ASP.NET TextBox spacing changed when using AntiXSSEncoder

Details

In .NET Framework 4.0, extra lines were inserted between lines of a multi-line text box on postback, if using the [System.Web.Security.AntiXss.AntiXssEncoder](#). In .NET Framework 4.5, those extra line breaks are not included, but only if the web app is targeting .NET Framework 4.5.

Suggestion

Be aware that 4.0 web apps retargeted to .NET Framework 4.5 may have multi-line text boxes improved to no longer insert extra line breaks. If this is not desirable, the app can have the old behavior when running on .NET Framework 4.5 by targeting the .NET Framework 4.0.

[+] Expand table

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

WebUtility.HtmlEncode and WebUtility.HtmlDecode round-trip BMP correctly

Details

For applications that target the .NET Framework 4.5, characters that are outside the Basic Multilingual Plane (BMP) round-trip correctly when they are passed to the [HtmlDecode\(String\)](#) methods.

Suggestion

This change should have no effect on current applications, but to restore the original behavior, set the `targetFramework` attribute of the `<httpRuntime>` element to a string other than "4.5". You can also set the `unicodeEncodingConformance` and `unicodeDecodingConformance` attributes of the `<webUtility>` configuration element to control this behavior independently of the targeted version of the .NET Framework.

[+] Expand table

Name	Value
Scope	Edge
Version	4.5
Type	Retargeting

Affected APIs

- [WebUtility.HtmlEncode\(String\)](#)
- [WebUtility.HtmlEncode\(String, TextWriter\)](#)

ClickOnce

Apps published with ClickOnce that use a SHA-256 code-signing certificate may fail on Windows 2003

Details

The executable is signed with SHA256. Previously, it was signed with SHA1 regardless of whether the code-signing certificate was SHA-1 or SHA-256. This applies to:

- All applications built with Visual Studio 2012 or later.
- Applications built with Visual Studio 2010 or earlier on systems with the .NET Framework 4.5 present. In addition, if the .NET Framework 4.5 or later is present, the ClickOnce manifest is also signed with SHA-256 for SHA-256 certificates regardless of the .NET Framework version against which it was compiled.

Suggestion

The change in signing the ClickOnce executable affects only Windows Server 2003 systems; they require that KB 938397 be installed. The change in signing the manifest

with SHA-256 even when an app targets the .NET Framework 4.0 or earlier versions introduces a runtime dependency on the .NET Framework 4.5 or a later version.

[\[\] Expand table](#)

Name	Value
Scope	Edge
Version	4.5
Type	Retargeting

Core

Foreach iterator variable is now scoped within the iteration, so closure capturing semantics are different (in C#5)

Details

Beginning with C# 5 (Visual Studio 2012), `foreach` iterator variables are scoped within the iteration. This can cause breaks if code was previously depending on the variables to not be included in the `foreach`'s closure. The symptom of this change is that an iterator variable passed to a delegate is treated as the value it has at the time the delegate is created, rather than the value it has at the time the delegate is invoked.

Suggestion

Ideally, code should be updated to expect the new compiler behavior. If the old semantics are required, the iterator variable can be replaced with a separate variable which is explicitly placed outside of the loop's scope.

[\[\] Expand table](#)

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

IAsyncResult.CompletedSynchronously property must be correct for the resulting task to complete

Details

When calling `TaskFactory.FromAsync`, the implementation of the `CompletedSynchronously` property must be correct for the resulting task to complete. That is, the property must return true if, and only if, the implementation completed synchronously. Previously, the property was not checked.

Suggestion

If `System.IAsyncResult` implementations correctly return true for the `System.IAsyncResult.CompletedSynchronously` property only when a task completed synchronously, then no break will be observed. Users should review `System.IAsyncResult` implementations they own (if any) to ensure that they correctly evaluate whether a task completed synchronously or not.

[+] Expand table

Name	Value
Scope	Edge
Version	4.5
Type	Retargeting

Affected APIs

- `TaskFactory.FromAsync(IAsyncResult, Action<IAsyncResult>)`
- `TaskFactory.FromAsync(IAsyncResult, Action<IAsyncResult>, TaskCreationOptions)`
- `TaskFactory.FromAsync(IAsyncResult, Action<IAsyncResult>, TaskCreationOptions, TaskScheduler)`
- `TaskFactory.FromAsync<TResult>(IAsyncResult, Func<IAsyncResult,TResult>)`
- `TaskFactory.FromAsync(Func<AsyncCallback,Object,IAsyncResult>, Action<IAsyncResult>, Object)`
- `TaskFactory.FromAsync(Func<AsyncCallback,Object,IAsyncResult>, Action<IAsyncResult>, Object, TaskCreationOptions)`
- `TaskFactory.FromAsync<TArg1>(Func<TArg1,AsyncCallback,Object,IAsyncResult>, Action<IAsyncResult>, TArg1, Object)`

- TaskFactory.FromAsync<TArg1>(Func<TArg1, AsyncCallback, Object, IAsyncResult>, Action<IAsyncResult>, TArg1, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TResult>(Func<AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, Object)
- TaskFactory.FromAsync<TResult>(Func<AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TResult>(IAsyncResult, Func<IAsyncResult, TResult>, TaskCreationOptions)
- TaskFactory.FromAsync<TResult>(IAsyncResult, Func<IAsyncResult, TResult>, TaskCreationOptions, TaskScheduler)
- TaskFactory.FromAsync<TArg1, TArg2>(Func<TArg1, TArg2, AsyncCallback, Object, IAsyncResult>, Action<IAsyncResult>, TArg1, TArg2, Object)
- TaskFactory.FromAsync<TArg1, TArg2>(Func<TArg1, TArg2, AsyncCallback, Object, IAsyncResult>, Action<IAsyncResult>, TArg1, TArg2, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TArg1, TResult>(Func<TArg1, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, Object)
- TaskFactory.FromAsync<TArg1, TResult>(Func<TArg1, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TArg1, TArg2, TResult>(Func<TArg1, TArg2, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, TArg2, Object)
- TaskFactory.FromAsync<TArg1, TArg2, TArg3>(Func<TArg1, TArg2, TArg3, AsyncCallback, Object, IAsyncResult>, Action<IAsyncResult>, TArg1, TArg2, TArg3, Object)
- TaskFactory.FromAsync<TArg1, TArg2, TArg3>(Func<TArg1, TArg2, TArg3, AsyncCallback, Object, IAsyncResult>, Action<IAsyncResult>, TArg1, TArg2, TArg3, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TArg1, TArg2, TResult>(Func<TArg1, TArg2, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, TArg2, Object, TaskCreationOptions)
- TaskFactory.FromAsync<TArg1, TArg2, TArg3, TResult>(Func<TArg1, TArg2, TArg3, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, TArg2, TArg3, Object)
- TaskFactory.FromAsync<TArg1, TArg2, TArg3, TResult>(Func<TArg1, TArg2, TArg3, AsyncCallback, Object, IAsyncResult>, Func<IAsyncResult, TResult>, TArg1, TArg2, TArg3, Object, TaskCreationOptions)

List<T>.ForEach can throw exception when modifying list item

Details

Beginning in .NET Framework 4.5, a [ForEach\(Action<T>\)](#) enumerator will throw an [System.InvalidOperationException](#) exception if an element in the calling collection is modified. Previously, this would not throw an exception but could lead to race conditions.

Suggestion

Ideally, code should be fixed to not modify lists while enumerating their elements because that is never a safe operation. To revert to the previous behavior, though, an app may target .NET Framework 4.0.

[\[\] Expand table](#)

Name	Value
Scope	Edge
Version	4.5
Type	Retargeting

Affected APIs

- [List<T>.ForEach\(Action<T>\)](#)

System.Uri parsing adheres to RFC 3987

Details

URI parsing has changed in several ways in .NET Framework 4.5. Note, however, that these changes only affect code targeting .NET Framework 4.5. If a binary targets .NET Framework 4.0, the old behavior will be observed. Changes to URI parsing in .NET Framework 4.5 include:

- URI parsing will perform normalization and character checking according to the latest IRI rules in RFC 3987.

- Unicode normalization form C will only be performed on the host portion of the URI.
- Invalid mailto: URIs will now cause an exception.
- Trailing dots at the end of a path segment are now preserved.
- `file://` URIs do not escape the `:` character.
- Unicode control characters `U+0080` through `U+009F` are not supported.
- Comma characters `,` or `%2c` are not automatically unescaped.

Suggestion

If the old .NET Framework 4.0 URI parsing semantics are necessary (they often aren't), they can be used by targeting .NET Framework 4.0. This can be accomplished by using a [System.Runtime.Versioning.TargetFrameworkAttribute](#) on the assembly, or through Visual Studio's project system UI in the 'project properties' page.

[+] Expand table

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

Affected APIs

- [Uri\(String\)](#)
- [Uri\(String, Boolean\)](#)
- [Uri\(String, UriKind\)](#)
- [Uri\(Uri, String\)](#)
- [Uri.TryCreate\(String, UriKind, Uri\)](#)
- [Uri.TryCreate\(Uri, String, Uri\)](#)
- [Uri.TryCreate\(Uri, Uri, Uri\)](#)

System.Uri.IsWellFormedUriString method returns false for relative URIs with a colon char in first segment

Details

Beginning with the .NET Framework 4.5, [IsWellFormedUriString\(String, UriKind\)](#) will treat relative URIs with a `:` in their first segment as not well formed. This is a change from [System.Uri.IsWellFormedUriString\(String, UriKind\)](#) behavior in the .NET Framework 4.0 that was made to conform to RFC3986.

Suggestion

This change (like many other URI changes) will only affect applications targeting the .NET Framework 4.5 (or later). To keep using the old behavior, target the app against the .NET Framework 4.0. Alternatively, scan URI's prior to calling [System.Uri.IsWellFormedUriString\(String, UriKind\)](#) looking for `:` characters that you may want to remove for validation purposes, if the old behavior is desirable.

[\[+\] Expand table](#)

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

Affected APIs

- [Uri.IsWellFormedUriString\(String, UriKind\)](#)

Entity Framework

Entity Framework version must match the .NET Framework version

Details

The Entity Framework (EF) version should be matched with the .NET Framework version. Entity Framework 5 is recommended for .NET Framework 4.5. There are some known issues with EF 4.x in a .NET Framework 4.5 project around [System.ComponentModel.DataAnnotations](#). In .NET Framework 4.5, these were moved to a different assembly, so there are issues determining which annotations to use.

Suggestion

Upgrade to Entity Framework 5 for .NET Framework 4.5

[] Expand table

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

Windows Forms

EncoderParameter ctor is obsolete

Details

The [EncoderParameter\(Encoder, Int32, Int32, Int32, Int32\)](#) constructor is obsolete now and will introduce build warnings if used.

Suggestion

Although the [EncoderParameter\(Encoder, Int32, Int32, Int32, Int32\)](#) constructor will continue to work, the following constructor should be used instead to avoid the obsolete build warning when re-compiling code with .NET Framework 4.5 tools: [EncoderParameter\(Encoder, Int32, EncoderParameterValueType, IntPtr\)](#).

[] Expand table

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

Affected APIs

- [EncoderParameter\(Encoder, Int32, Int32, Int32, Int32\)](#)

Windows Communication Foundation (WCF)

Writing binary output using BodyWriter

Details

If you're deriving from the class `System.ServiceModel.Channels.BodyWriter` and using the implementation of `OnWriteBodyContents(XmlDictionaryWriter writer)` to write binary output, some changes may need to be made when you retarget to .NET Framework 4.5. Check the write state, and if it's `WriterState.Start`, emit the `Binary` wrapping XML element as shown in the following code snippet.

C#

```
protected override void OnWriteBodyContents(XmlDictionaryWriter writer)
{
    bool wroteStartElement = false;
    if (writer.WriteState == WriteState.Start)
    {
        writer.WriteStartElement("Binary", string.Empty);
        wroteStartElement = true;
    }
    writer.WriteBase64(buffer, offset, count);
    if (wroteStartElement)
    {
        writer.WriteEndElement();
    }
}
```

In addition, if you're deriving from the class

`System.ServiceModel.Channels.StreamBodyWriter` and have overridden the method `OnWriteBodyContents(XmlDictionaryWriter writer)`, some changes may be required. When targeting .NET Framework 4.0, it was necessary to explicitly write the `Binary` element when overriding this method. This is no longer needed when you target .NET Framework 4.5, and doing so causes the body to not be written.

Windows Presentation Foundation (WPF)

WPF TextBox.Text can be out-of-sync with databinding

Details

In some cases, the [Text](#) property reflects a previous value of the databound property value if the property is modified during a databinding write operation.

Suggestion

This should have no negative impact. However, you can restore the previous behavior by setting the [KeepTextBoxDisplaySynchronizedWithTextProperty](#) property to `false`.

[+] [Expand table](#)

Value	
Scope	Edge
Version	4.5
Type	Retargeting

Affected APIs

- [TextBox.Text](#)

Windows Workflow Foundation (WF)

New (ambiguous) `Dispatcher.Invoke` overloads could result in different behavior

Details

The .NET Framework 4.5 adds new overloads to [Dispatcher.Invoke](#) that include a parameter of type [Action](#). When existing code is recompiled, compilers may resolve calls to `Dispatcher.Invoke` methods that have a [Delegate](#) parameter as calls to `Dispatcher.Invoke` methods with an [Action](#) parameter. If a call to a `Dispatcher.Invoke` overload with a [Delegate](#) parameter is resolved as a call to a `Dispatcher.Invoke` overload with an [Action](#) parameter, the following differences in behavior may occur:

- If an exception occurs, the [UnhandledExceptionFilter](#) and [UnhandledException](#) events are not raised. Instead, exceptions are handled by the [System.Threading.Tasks.TaskScheduler.UnobservedTaskException](#) event.
- Calls to some members, such as [Result](#), block until the operation has completed.

Suggestion

To avoid ambiguity (and potential differences in exception handling or blocking behaviors), code calling Dispatcher.Invoke can pass an empty object[] as a second parameter to the Invoke call to be sure of resolving to the .NET Framework 4.0 method overload.

[+] Expand table

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

Affected APIs

- [Dispatcher.Invoke\(Delegate, Object\[\]\)](#)
- [Dispatcher.Invoke\(Delegate, TimeSpan, Object\[\]\)](#)
- [Dispatcher.Invoke\(Delegate, TimeSpan, DispatcherPriority, Object\[\]\)](#)
- [Dispatcher.Invoke\(Delegate, DispatcherPriority, Object\[\]\)](#)

Some WorkFlow drag-and-drop APIs are obsolete

Details

This WorkFlow drag-and-drop API is obsolete and will cause compiler warnings if the app is rebuilt against 4.5.

Suggestion

New [System.Activities.Presentation.DragDropHelper](#) APIs that support operations with multiple objects should be used instead. Alternatively, the build warnings can be suppressed or they can be avoided by using an older compiler. The APIs are still supported.

[+] Expand table

Name	Value
Scope	Minor

Name	Value
Type	Retargeting

Affected APIs

- DragDropHelper.DoDragMove(WorkflowViewElement, Point)
- DragDropHelper.GetCompositeView(DragEventArgs)
- DragDropHelper.GetDraggedModelItem(DragEventArgs)
- DragDropHelper.GetDroppedObject(DependencyObject, DragEventArgs, EditingContext)

WorkFlow 3.0 types are obsolete

Details

Windows Workflow Foundation (WWF) 3.0 APIs (those from the System.Workflow namespace) are now obsolete.

Suggestion

New WWF 4.0 APIs (in System.Activities) should be used instead. An example of using the new APIs can be found [here](#) and further guidance is available [here](#). Alternatively, since the WWF 3.0 APIs are still supported, they may be used and the build-time warning avoided either by suppressing it or by using an older compiler.

[+] Expand table

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

WorkflowDesigner.Load doesn't remove symbol property

Details

When targeting the .NET Framework 4.5 in the workflow designer, and loading a re-hosted 3.5 workflow with the [Load\(\)](#) method, a [System.Xaml.XamlDuplicateMemberException](#) is thrown while saving the workflow.

Suggestion

This bug only manifests when targeting .NET Framework 4.5 in the workflow designer, so it can be worked around by setting the `WorkflowDesigner.Context.Services.GetService<DesignerConfigurationService>().TargetFrameworkName` to the 4.0 .NET Framework.

Alternatively, the issue may be avoided by using the [Load\(String\)](#) method to load the workflow, instead of [Load\(\)](#).

[+] Expand table

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

Affected APIs

- [WorkflowDesigner.Load\(\)](#)

XML, XSLT

XML schema validation is stricter

Details

In the .NET Framework 4.5, XML schema validation is more strict. If you use xsd:anyURI to validate a URI such as a mailto protocol, validation fails if there are spaces in the URI. In previous versions of the .NET Framework, validation succeeded. The change affects only applications that target the .NET Framework 4.5.

Suggestion

If looser .NET Framework 4.0 validation is needed, the validating application can target version 4.0 of the .NET Framework. When retargeting to .NET Framework 4.5, however, code review should be done to be sure that invalid URLs (with spaces) are not expected as attribute values with the anyURI data type.

[+] Expand table

Name	Value
Scope	Minor
Version	4.5
Type	Retargeting

.NET Framework 4.5.1

ADO.NET

DbParameter.Precision and DbParameter.Scale are now public virtual members

Details

Precision and Scale are implemented as public virtual properties. They replace the corresponding explicit interface implementations, [IDbDataParameter.Precision](#) and [IDbDataParameter.Scale](#).

Suggestion

When re-building an ADO.NET database provider, these differences will require the 'override' keyword to be applied to the Precision and Scale properties. This is only needed when re-building the components; existing binaries will continue to work.

[+] Expand table

Name	Value
Scope	Minor
Version	4.5.1

Name	Value
Type	Retargeting

Affected APIs

- [DbParameter.Precision](#)
- [DbParameter.Scale](#)

Core

ObsoleteAttribute exports as both ObsoleteAttribute and DeprecatedAttribute in WinMD scenarios

Details

When you create a Windows Metadata library (.winmd file), the [System.ObsoleteAttribute](#) attribute is exported as both [System.ObsoleteAttribute](#) and [Windows.Foundation.DeprecatedAttribute](#).

Suggestion

Recompilation of existing source code that uses the [System.ObsoleteAttribute](#) attribute may generate warnings when consuming that code from C++/CX or JavaScript. We do not recommend applying both [System.ObsoleteAttribute](#) and [Windows.Foundation.DeprecatedAttribute](#) to code in managed assemblies; it may result in build warnings.

[\[+\] Expand table](#)

Name	Value
Scope	Edge
Version	4.5.1
Type	Retargeting

Entity Framework

Building an Entity Framework edmx with Visual Studio 2013 can fail with error MSB4062 if using the EntityDeploySplit or EntityClean tasks

Details

MSBuild 12.0 tools (included in Visual Studio 2013) changed MSBuild file locations, causing older Entity Framework targets files to be invalid. The result is that `EntityDeploySplit` and `EntityClean` tasks fail because they are unable to find `Microsoft.Data.Entity.Build.Tasks.dll`. Note that this break is because of a toolset (MSBuild/VS) change, not because of a .NET Framework change. It will only occur when upgrading developer tools, not when merely upgrading the .NET Framework.

Suggestion

Entity Framework targets files are fixed to work with the new MSBuild layout beginning in the .NET Framework 4.6. Upgrading to that version of the Framework will fix this issue. Alternatively, [this workaround ↗](#) can be used to patch the targets files directly.

 Expand table

Name	Value
Scope	Major
Version	4.5.1
Type	Retargeting

MSBuild

ResolveAssemblyReference task now warns of dependencies with the wrong architecture

Details

The task emits a warning, MSB3270, which indicates that a reference or any of its dependencies does not match the app's architecture. For example, this occurs if an app that was compiled with the `AnyCPU` option includes an `x86` reference. Such a scenario

could result in an app failure at run time (in this case, if the app is deployed as an x64 process).

Suggestion

There are two areas of impact:

- Recompilation generates warnings that did not appear when the app was compiled under a previous version of MSBuild. However, because the warning identifies a possible source of runtime failure, it should be investigated and addressed.
- If warnings are treated as errors, the app will fail to compile.

[] Expand table

Name	Value
Scope	Minor
Version	4.5.1
Type	Retargeting

Windows Presentation Foundation (WPF)

Two-way data-binding to a property with a non-public setter is not supported

Details

Attempting to data bind to a property without a public setter has never been a supported scenario. Beginning in the .NET Framework 4.5.1, this scenario will throw an [System.InvalidOperationException](#). Note that this new exception will only be thrown for apps that specifically target the .NET Framework 4.5.1. If an app targets the .NET Framework 4.5, the call will be allowed. If the app does not target a particular .NET Framework version, the binding will be treated as one-way.

Suggestion

The app should be updated to either use one-way binding, or expose the property's setter publicly. Alternatively, targeting the .NET Framework 4.5 will cause the app to exhibit the old behavior.

[] Expand table

Name	Value
Scope	Minor
Version	4.5.1
Type	Retargeting

Affected APIs

- [BindingMode.TwoWay](#)

.NET Framework 4.5.2

Visual Basic .NET

VB.NET no longer supports partial namespace qualification for System.Windows APIs

Details

Beginning in .NET Framework 4.5.2, VB.NET projects cannot specify System.Windows APIs with partially-qualified namespaces. For example, referring to `Windows.Forms.DialogResult` will fail. Instead, code must refer to the fully qualified name ([DialogResult](#)) or import the specific namespace and refer simply to [System.Windows.Forms.DialogResult](#).

Suggestion

Code should be updated to refer to `System.Windows` APIs either with simple names (and importing the relevant namespace) or with fully qualified names.

[] Expand table

Name	Value
Scope	Minor
Version	4.5.2

Name	Value
Type	Retargeting

Windows Forms

DataObject.GetData now retrieves data as UTF-8

Details

For apps that target the .NET Framework 4 or that run on the .NET Framework 4.5.1 or earlier versions, `DataObject.GetData` retrieves HTML-formatted data as an ASCII string. As a result, non-ASCII characters (characters whose ASCII codes are greater than 0x7F) are represented by two random characters.

For apps that target the .NET Framework 4.5 or later and run on the .NET Framework 4.5.2, `DataObject.GetData` retrieves HTML-formatted data as UTF-8, which represents characters greater than 0x7F correctly.

Suggestion

If you implemented a workaround for the encoding problem with HTML-formatted strings (for example, by explicitly encoding the HTML string retrieved from the Clipboard by passing it to `System.Text.UTF8Encoding.GetString(Byte[], Int32, Int32)`) and you're retargeting your app from version 4 to 4.5, that workaround should be removed. If the old behavior is needed for some reason, the app can target the .NET Framework 4.0 to get that behavior.

 Expand table

Name	Value
Scope	Edge
Version	4.5.2
Type	Retargeting

Affected APIs

- [DataObject.GetData\(String\)](#)

- [DataObject.GetData\(Type\)](#)
- [DataObject.GetData\(String, Boolean\)](#)

Windows Workflow Foundation (WF)

WorkflowDesigner.Load doesn't remove symbol property

Details

When targeting the .NET Framework 4.5 in the workflow designer, and loading a re-hosted 3.5 workflow with the [Load\(\)](#) method, a [System.Xaml.XamlDuplicateMemberException](#) is thrown while saving the workflow.

Suggestion

This bug only manifests when targeting .NET Framework 4.5 in the workflow designer, so it can be worked around by setting the

```
WorkflowDesigner.Context.Services.GetService<DesignerConfigurationService>()
().TargetFrameworkName
```

 to the 4.0 .NET Framework.

Alternatively, the issue may be avoided by using the [Load\(String\)](#) method to load the workflow, instead of [Load\(\)](#).

[+] [Expand table](#)

Name	Value
Scope	Major
Version	4.5
Type	Retargeting

Affected APIs

- [WorkflowDesigner.Load\(\)](#)

 Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

Retargeting changes for migration to .NET Framework 4.6.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.6](#), [4.6.1](#), and [4.6.2](#).

.NET Framework 4.6

ASP.NET

HtmlTextWriter does not render `
` element correctly

Details

Beginning in the .NET Framework 4.6, calling `RenderBeginTag(String)` and `RenderEndTag()` with a `
` element will correctly insert only one `
` (instead of two)

Suggestion

If an app depended on the extra `
` tag, `RenderBeginTag(String)` should be called a second time. Note that this behavior change only affects apps that target the .NET Framework 4.6 or later, so another option is to target a previous version of the .NET Framework in order to get the old behavior.

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

Affected APIs

- `HtmlTextWriter.RenderBeginTag(String)`
- `HtmlTextWriter.RenderBeginTag(HtmlTextWriterTag)`

ClickOnce

Apps published with ClickOnce that use a SHA-256 code-signing certificate may fail on Windows 2003

Details

The executable is signed with SHA256. Previously, it was signed with SHA1 regardless of whether the code-signing certificate was SHA-1 or SHA-256. This applies to:

- All applications built with Visual Studio 2012 or later.
- Applications built with Visual Studio 2010 or earlier on systems with the .NET Framework 4.5 present. In addition, if the .NET Framework 4.5 or later is present, the ClickOnce manifest is also signed with SHA-256 for SHA-256 certificates regardless of the .NET Framework version against which it was compiled.

Suggestion

The change in signing the ClickOnce executable affects only Windows Server 2003 systems; they require that KB 938397 be installed. The change in signing the manifest with SHA-256 even when an app targets the .NET Framework 4.0 or earlier versions introduces a runtime dependency on the .NET Framework 4.5 or a later version.

Name	Value
Scope	Edge
Version	4.5
Type	Retargeting

ClickOnce supports SHA-256 on 4.0-targeted apps

Details

Previously, a ClickOnce app with a certificate signed with SHA-256 would require .NET Framework 4.5 or later to be present, even if the app targeted 4.0. Now, .NET Framework 4.0-targeted ClickOnce apps can run on .NET Framework 4.0, even if signed with SHA-256.

Suggestion

This change removes that dependency and allows SHA-256 certificates to be used to sign ClickOnce apps that target .NET Framework 4 and earlier versions.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Core

CurrentCulture and CurrentUICulture flow across tasks

Details

Beginning in the .NET Framework 4.6, [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) are stored in the thread's [System.Threading.ExecutionContext](#), which flows across asynchronous operations. This means that changes to [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) will be reflected in tasks which are later run asynchronously. This is different from the behavior of previous .NET Framework versions (which would reset [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) in all asynchronous tasks).

Suggestion

Apps affected by this change may work around it by explicitly setting the desired [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) as the first operation in an async Task. Alternatively, the old behavior (of not flowing [System.Globalization.CultureInfo.CurrentCulture](#)/[System.Globalization.CultureInfo.CurrentUICulture](#)) may be opted into by setting the following compatibility switch:

C#

```
AppContext.SetSwitch("Switch.System.Globalization.NoAsyncCurrentCulture",  
true);
```

This issue has been fixed by WPF in .NET Framework 4.6.2. It has also been fixed in .NET Frameworks 4.6, 4.6.1 through [KB 3139549](#). Applications targeting .NET Framework 4.6 or later will automatically get the right behavior in WPF applications - `System.Globalization.CultureInfo.CurrentCulture/System.Globalization.CultureInfo.CurrentUICulture` would be preserved across Dispatcher operations.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- `CultureInfo.CurrentCulture`
- `Thread.CurrentCulture`
- `CultureInfo.CurrentUICulture`
- `Thread.CurrentUICulture`

ETW event names cannot differ only by a "Start" or "Stop" suffix

Details

In the .NET Framework 4.6 and 4.6.1, the runtime throws an [ArgumentException](#) when two Event Tracing for Windows (ETW) event names differ only by a "Start" or "Stop" suffix (as when one event is named `LogUser` and another is named `LogUserStart`). In this case, the runtime cannot construct the event source, which cannot emit any logging.

Suggestion

To prevent the exception, ensure that no two event names differ only by a "Start" or "Stop" suffix. This requirement is removed starting with the .NET Framework 4.6.2; the runtime can disambiguate event names that differ only by the "Start" and "Stop" suffix.

Name	Value
Scope	Edge
Version	4.6

Name	Value
Type	Retargeting

Entity Framework

Building an Entity Framework edmx with Visual Studio 2013 can fail with error MSB4062 if using the EntityDeploySplit or EntityClean tasks

Details

MSBuild 12.0 tools (included in Visual Studio 2013) changed MSBuild file locations, causing older Entity Framework targets files to be invalid. The result is that `EntityDeploySplit` and `EntityClean` tasks fail because they are unable to find `Microsoft.Data.Entity.Build.Tasks.dll`. Note that this break is because of a toolset (MSBuild/VS) change, not because of a .NET Framework change. It will only occur when upgrading developer tools, not when merely upgrading the .NET Framework.

Suggestion

Entity Framework targets files are fixed to work with the new MSBuild layout beginning in the .NET Framework 4.6. Upgrading to that version of the Framework will fix this issue. Alternatively, [this workaround ↗](#) can be used to patch the targets files directly.

Name	Value
Scope	Major
Version	4.5.1
Type	Retargeting

JIT

IL ret not allowed in a try region

Details

Unlike the JIT64 just-in-time compiler, RyuJIT (used in .NET Framework 4.6) does not allow an IL ret instruction in a try region. Returning from a try region is disallowed by the ECMA-335 specification, and no known managed compiler generates such IL. However, the JIT64 compiler will execute such IL if it is generated using reflection emit.

Suggestion

If an app is generating IL that includes a ret opcode in a try region, the app may target .NET Framework 4.5 to use the old JIT and avoid this break. Alternatively, the generated IL may be updated to return after the try region.

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

New 64-bit JIT compiler in the .NET Framework 4.6

Details

Starting with the .NET Framework 4.6, a new 64-bit JIT compiler is used for just-in-time compilation. In some cases, an unexpected exception is thrown or a different behavior is observed than if an app is run using the 32-bit compiler or the older 64-bit JIT compiler. This change does not affect the 32-bit JIT compiler. The known differences include the following:

- Under certain conditions, an unboxing operation may throw a [NullReferenceException](#) in Release builds with optimization turned on.
- In some cases, execution of production code in a large method body may throw a [StackOverflowException](#).
- Under certain conditions, structures passed to a method are treated as reference types rather than as value types in Release builds. One of the manifestations of this issue is that the individual items in a collection appear in an unexpected order.
- Under certain conditions, the comparison of [UInt16](#) values with their high bit set is incorrect if optimization is enabled.
- Under certain conditions, particularly when initializing array values, memory initialization by the [OpCodes.Initblk](#) IL instruction may initialize memory with an incorrect value. This can result either in an unhandled exception or incorrect output.

- Under certain rare conditions, a conditional bit test can return the incorrect [Boolean](#) value or throw an exception if compiler optimizations are enabled.
- Under certain conditions, if an `if` statement is used to test for a condition before entering a `try` block and in the exit from the `try` block, and the same condition is evaluated in the `catch` or `finally` block, the new 64-bit JIT compiler removes the `if` condition from the `catch` or `finally` block when it optimizes code. As a result, code inside the `if` statement in the `catch` or `finally` block is executed unconditionally.

Suggestion

Mitigation of known issues

If you encounter the issues listed above, you can address them by doing any of the following:

- Upgrade to the .NET Framework 4.6.2. The new 64-bit compiler included with the .NET Framework 4.6.2 addresses each of these known issues.
- Ensure that your version of Windows is up to date by running Windows Update. Service updates to the .NET Framework 4.6 and 4.6.1 address each of these issues except the [NullReferenceException](#) in an unboxing operation.
- Compile with the older 64-bit JIT compiler. See the [Mitigation of other issues](#) section for more information on how to do this. **Mitigation of other issues**
If you encounter any other difference in behavior between code compiled with the older 64-bit compiler and the new 64-bit JIT compiler, or between the debug and release versions of your app that are both compiled with the new 64-bit JIT compiler, you can do the following to compile your app with the older 64-bit JIT compiler:
 - On a per-application basis, you can add the `<` element to your application's configuration file. The following disables compilation with the new 64-bit JIT compiler and instead uses the legacy 64-bit JIT compiler.

XML

```
<?xml version ="1.0"?>
<configuration>
  <runtime>
    <useLegacyJit enabled="1" />
  </runtime>
</configuration>
```

- On a per-user basis, you can add a `REG_DWORD` value named `useLegacyJit` to the `HKEY_CURRENT_USER\Software\Microsoft\.NETFramework` key of the registry. A value of 1 enables the legacy 64-bit JIT compiler; a value of 0 disables it and enables the new 64-bit JIT compiler.
- On a per-machine basis, you can add a `REG_DWORD` value named `useLegacyJit` to the `HKEY_LOCAL_MACHINE\Software\Microsoft\.NETFramework` key of the registry. A value of 1 enables the legacy 64-bit JIT compiler; a value of 0 disables it and enables the new 64-bit JIT compiler. You can also let us know about the problem by reporting a bug on [Microsoft Connect](#).

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

Networking

Certificate EKU OID validation

Details

Starting with .NET Framework 4.6, the `SslStream` or `ServicePointManager` classes perform enhanced key use (EKU) object identifier (OID) validation. An enhanced key usage (EKU) extension is a collection of object identifiers (OIDs) that indicate the applications that use the key. EKU OID validation uses remote certificate callbacks to ensure that the remote certificate has the correct OIDs for the intended purpose.

Suggestion

If this change is undesirable, you can disable certificate EKU OID validation by adding the following switch to the `<AppContextSwitchOverrides>` in the `app.config` of your app configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides>
```

```
value="Switch.System.Net.DontCheckCertificateEKUs=true" />  
</runtime>
```

ⓘ Important

This setting is provided for backward compatibility only. Its use is otherwise not recommended.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- [System.Net.Security.SslStream](#)
- [System.Net.ServicePointManager](#)
- [System.Net.Http.HttpClient](#)
- [System.Net.Mail.SmtpClient](#)
- [System.Net.HttpWebRequest](#)
- [System.Net.FtpWebRequest](#)

Only Tls 1.0, 1.1 and 1.2 protocols supported in [System.Net.ServicePointManager](#) and [System.Net.Security.SslStream](#)

Details

Starting with the .NET Framework 4.6, the [ServicePointManager](#) and [SslStream](#) classes are only allowed to use one of the following three protocols: Tls1.0, Tls1.1, or Tls1.2. The SSL3.0 protocol and RC4 cipher are not supported.

Suggestion

The recommended mitigation is to upgrade the sever-side app to Tls1.0, Tls1.1, or Tls1.2. If this is not feasible, or if client apps are broken, the [System.AppContext](#) class can be used to opt out of this feature in either of two ways:

- By programmatically setting compat switches on the [System.AppContext](#), as explained [here ↗](#).
- By adding the following line to the `<runtime>` section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Net.DontEnableSchUseStrongCrypto=true"/>
```

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- `SecurityProtocolType.Ssl3`
- `SslProtocols.None`
- `SslProtocols.Ssl2`
- `SslProtocols.Ssl3`

TLS 1.x by default passes the SCH_SEND_AUX_RECORD flag to the underlying SCHANNEL API

Details

When using TLS 1.x, the .NET Framework relies on the underlying Windows SCHANNEL API. Starting with .NET Framework 4.6, the `SCH_SEND_AUX_RECORD` flag is passed by default to SCHANNEL. This causes SCHANNEL to split data to be encrypted into two separate records, the first as a single byte and the second as $n-1$ bytes. In rare cases, this breaks communication between clients and existing servers that make the assumption that the data resides in a single record.

Suggestion

If this change breaks communication with an existing server, you can disable sending the `SCH_SEND_AUX_RECORD` flag and restore the previous behavior of not splitting data into separate records by adding the following switch to the

<AppContextSwitchOverrides> element in the <runtime> section of your app configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Net.DontEnableSchSendAuxRecord=true" />
</runtime>
```

ⓘ Important

This setting is provided for backward compatibility only. Its use is otherwise not recommended.

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

Affected APIs

- [System.Net.Security.SslStream](#)
- [System.Net.ServicePointManager](#)
- [System.Net.Http.HttpClient](#)
- [System.Net.Mail.SmtpClient](#)
- [System.Net.HttpWebRequest](#)
- [System.Net.FtpWebRequest](#)

Windows Communication Foundation (WCF)

Calling CreateDefaultAuthorizationContext with a null argument has changed

Details

The implementation of the [System.IdentityModel.Policy.AuthorizationContext](#) returned by a call to the

`System.IdentityModel.Policy.AuthorizationContext.CreateDefaultAuthorizationContext(IList<IAuthorizationPolicy>)` with a null authorizationPolicies argument has changed its implementation in the .NET Framework 4.6.

Suggestion

In rare cases, WCF apps that use custom authentication may see behavioral differences. In such cases, the previous behavior can be restored in either of two ways:

- Recompile your app to target an earlier version of the .NET Framework than 4.6. For IIS-hosted services, use the `<httpRuntime targetFramework="x.x">` element to target an earlier version of the .NET Framework.
- Add the following line to the `<appSettings>` section of your app.config file:

XML

```
<add  
key="appContext.SetSwitch:Switch.System.IdentityModel.EnableCachedEmpty  
DefaultAuthorizationContext" value="true" />
```

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- `AuthorizationContext.CreateDefaultAuthorizationContext(IList<IAuthorizationPolicy>)`

Windows Forms

Icon.ToBitmap successfully converts icons with PNG frames into Bitmap objects

Details

Starting with the apps that target the .NET Framework 4.6, the [Icon.ToBitmap](#) method successfully converts icons with PNG frames into Bitmap objects.

In apps that target the .NET Framework 4.5.2 and earlier versions, the [Icon.ToBitmap](#) method throws an [ArgumentOutOfRangeException](#) exception if the Icon object has PNG frames.

This change affects apps that are recompiled to target the .NET Framework 4.6 and that implement special handling for the [ArgumentOutOfRangeException](#) that is thrown when an Icon object has PNG frames. When running under the .NET Framework 4.6, the conversion is successful, an [ArgumentOutOfRangeException](#) is no longer thrown, and therefore the exception handler is no longer invoked.

Suggestion

If this behavior is undesirable, you can retain the previous behavior by adding the following element to the `<runtime>` section of your app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Drawing.DontSupportPngFramesInIcons=true" />
```

If the app.config file already contains the `AppContextSwitchOverrides` element, the new value should be merged with the value attribute like this:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Drawing.DontSupportPngFramesInIcons=true;<previous  
key>=<previous value>" />
```

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- [Icon.ToBitmap\(\)](#)

Windows Presentation Foundation (WPF)

CurrentCulture is not preserved across WPF Dispatcher operations

Details

Beginning in the .NET Framework 4.6, changes to [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) made within a [System.Windows.Threading.Dispatcher](#) will be lost at the end of that dispatcher operation. Similarly, changes to [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) made outside of a Dispatcher operation may not be reflected when that operation executes. Practically speaking, this means that [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) changes may not flow between WPF UI callbacks and other code in a WPF application. This is due to a change in [System.Threading.ExecutionContext](#) that causes [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) to be stored in the execution context beginning with apps targeting the .NET Framework 4.6. WPF dispatcher operations store the execution context used to begin the operation and restore the previous context when the operation is completed. Because [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) are now part of that context, changes to them within a dispatcher operation are not persisted outside of the operation.

Suggestion

Apps affected by this change may work around it by storing the desired [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) in a field and checking in all Dispatcher operation bodies (including UI event callback handlers) that the correct [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) are set. Alternatively, because the [ExecutionContext](#) change underlying this WPF change only affects apps targeting the .NET Framework 4.6 or newer, this break can be avoided by targeting the .NET Framework 4.5.2. Apps that target .NET Framework 4.6 or later can also work around this by setting the following compatibility switch:

C#

```
ApplicationContext.SetSwitch("Switch.System.Globalization.NoAsyncCurrentCulture",  
true);
```

This issue has been fixed by WPF in .NET Framework 4.6.2. It has also been fixed in .NET Frameworks 4.6, 4.6.1 through [KB 3139549](#). Applications targeting .NET Framework 4.6 or later will automatically get the right behavior in WPF applications - [System.Globalization.CultureInfo.CurrentCulture](#)/[System.Globalization.CultureInfo.CurrentUICulture](#) would be preserved across Dispatcher operations.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

WPF layout rounding of margins has changed

Details

The way in which margins are rounded and borders and the background inside of them has changed. As a result of this change:

- The width or height of elements may grow or shrink by at most one pixel.
- The placement of an object can move by at most one pixel.
- Centered elements can be vertically or horizontally off center by at most one pixel.
By default, this new layout is enabled only for apps that target the .NET Framework 4.6.

Suggestion

Since this modification tends to eliminate clipping of the right or bottom of WPF controls at high DPIs, apps that target earlier versions of the .NET Framework but are running on the .NET Framework 4.6 can opt into this new behavior by adding the following line to the `<runtime>` section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.MS.Internal.DoNotApplyLayoutRoundingToMarginsAndBorderThickness"/>
```

```
s=false" />'
```

Apps that target the .NET Framework 4.6 but want WPF controls to render using the previous layout algorithm can do so by adding the following line to the `<runtime>` section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.MS.Internal.DoNotApplyLayoutRoundingToMarginsAndBorderThickness  
s=true" />.
```

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

XML, XSLT

XmlWriter throws on invalid surrogate pairs

Details

For apps that target the .NET Framework 4.5.2 or previous versions, writing an invalid surrogate pair using exception fallback handling does not always throw an exception. For apps that target the .NET Framework 4.6, attempting to write an invalid surrogate pair throws an [System.ArgumentException](#).

Suggestion

If necessary, this break can be avoided by targeting the .NET Framework 4.5.2 or earlier. Alternatively, invalid surrogate pairs can be pre-processed into valid xml prior to writing them.

Name	Value
Scope	Edge
Version	4.6

Name	Value
Type	Retargeting

Affected APIs

- [XmlWriter.WriteString\(String\)](#)
- [XmlWriter.WriteString\(String, String\)](#)
- [XmlWriter.WriteString\(String, String, String\)](#)
- [XmlWriter.WriteStringAsync\(String, String, String\)](#)
- [XmlWriter.WriteCData\(String\)](#)
- [XmlWriter.WriteCDataAsync\(String\)](#)
- [XmlWriter.WriteChars\(Char\[\], Int32, Int32\)](#)
- [XmlWriter.WriteCharsAsync\(Char\[\], Int32, Int32\)](#)
- [XmlWriter.WriteComment\(String\)](#)
- [XmlWriter.WriteCommentAsync\(String\)](#)
- [XmlWriter.WriteEntityRef\(String\)](#)
- [XmlWriter.WriteEntityRefAsync\(String\)](#)
- [XmlWriter.WriteRaw\(Char\[\], Int32, Int32\)](#)
- [XmlWriter.WriteProcessingInstruction\(String, String\)](#)
- [XmlWriter.WriteProcessingInstructionAsync\(String, String\)](#)
- [XmlWriter.WriteRaw\(String\)](#)
- [XmlWriter.WriteRawAsync\(Char\[\], Int32, Int32\)](#)
- [XmlWriter.WriteRawAsync\(String\)](#)
- [XmlWriter.WriteString\(String\)](#)
- [XmlWriter.WriteStringAsync\(String\)](#)
- [XmlWriter.WriteSurrogateCharEntity\(Char, Char\)](#)
- [XmlWriter.WriteSurrogateCharEntityAsync\(Char, Char\)](#)
- [XmlWriter.WriteLine\(String\)](#)

XSD Schema validation now correctly detects violations of unique constraints if compound keys are used and one key is empty

Details

Versions of the .NET Framework prior to 4.6 had a bug that caused XSD validation to not detect unique constraints on compound keys if one of the keys was empty. In the .NET Framework 4.6, this issue is corrected. This will result in more correct validation, but it may also result in some XML not validating which previously would have.

Suggestion

If looser .NET Framework 4.0 validation is needed, the validating application can target version 4.5 (or earlier) of the .NET Framework. When retargeting to .NET Framework 4.6, however, code review should be done to be sure that duplicate compound keys (as described in this issue's description) are not expected to validate.

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

.NET Framework 4.6.1

Core

Change in path separator character in `FullName` property of `ZipArchiveEntry` objects

Details

For apps that target the .NET Framework 4.6.1 and later versions, the path separator character has changed from a backslash ("\") to a forward slash ("/") in the `FullName` property of `ZipArchiveEntry` objects created by overloads of the `CreateFromDirectory` method. The change brings the .NET implementation into conformity with section 4.4.17.1 of the [ZIP File Format Specification](#) and allows .ZIP archives to be decompressed on non-Windows systems.

Decompressing a zip file created by an app that targets a previous version of the .NET Framework on non-Windows operating systems such as the Macintosh fails to preserve the directory structure. For example, on the Macintosh, it creates a set of files whose filename concatenates the directory path, along with any backslash ("\") characters, and the filename. As a result, the directory structure of decompressed files is not preserved.

Suggestion

The impact of this change on .ZIP files that are decompressed on the Windows operating system by APIs in the .NET Framework `System.IO` namespace should be

minimal, since these APIs can seamlessly handle either a forward slash ("/") or a backslash ("\") as the path separator character.

If this change is undesirable, you can opt out of it by adding a configuration setting to the `<runtime>` section of your application configuration file. The following example shows both the `<runtime>` section and the

`Switch.System.IO.Compression.ZipFile.UseBackslash` opt-out switch:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.Compression.ZipFile.UseBackslash=true" />
</runtime>
```

In addition, apps that target previous versions of the .NET Framework but are running on the .NET Framework 4.6.1 and later versions can opt in to this behavior by adding a configuration setting to the `<runtime>` section of the application configuration file. The following shows both the `<runtime>` section and the

`Switch.System.IO.Compression.ZipFile.UseBackslash` opt-in switch.

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.Compression.ZipFile.UseBackslash=false" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.6.1
Type	Retargeting

Affected APIs

- `ZipFile.CreateDirectory(String, String)`
- `ZipFile.CreateDirectory(String, String, CompressionLevel, Boolean)`
- `ZipFile.CreateDirectory(String, String, CompressionLevel, Boolean, Encoding)`

Windows Communication Foundation (WCF)

WCF binding with the TransportWithMessageCredential security mode

Details

Beginning in the .NET Framework 4.6.1, WCF binding that uses the `TransportWithMessageCredential` security mode can be set up to receive messages with unsigned "to" headers for asymmetric security keys. By default, unsigned "to" headers will continue to be rejected in .NET Framework 4.6.1. They will only be accepted if an application opts into this new mode of operation using the `Switch.System.ServiceModel.AllowUnsignedToHeader` configuration switch.

Suggestion

Because this is an opt-in feature, it should not affect the behavior of existing apps. To control whether the new behavior is used or not, use the following configuration setting:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.ServiceModel.AllowUnsignedToHeader=true" />
</runtime>
```

Name	Value
Scope	Transparent
Version	4.6.1
Type	Retargeting

Affected APIs

- `BasicHttpSecurityMode.TransportWithMessageCredential`
- `BasicHttpsSecurityMode.TransportWithMessageCredential`
- `SecurityMode.TransportWithMessageCredential`
- `WSFederationHttpSecurityMode.TransportWithMessageCredential`

X509CertificateClaimSet.FindClaims Considers All claimTypes

Details

In apps that target the .NET Framework 4.6.1, if an X509 claim set is initialized from a certificate that has multiple DNS entries in its SAN field, the [System.IdentityModel.Claims.X509CertificateClaimSet.FindClaims\(String, String\)](#) method attempts to match the claimType argument with all the DNS entries. For apps that target previous versions of the .NET Framework, the [System.IdentityModel.Claims.X509CertificateClaimSet.FindClaims\(String, String\)](#) method attempts to match the claimType argument only with the last DNS entry.

Suggestion

This change only affects applications targeting the .NET Framework 4.6.1. This change may be disabled (or enabled if targeting pre-4.6.1) with the [DisableMultipleDNSEntries](#) compatibility switch.

Name	Value
Scope	Minor
Version	4.6.1
Type	Retargeting

Affected APIs

- [X509CertificateClaimSet.FindClaims\(String, String\)](#)

Windows Forms

Application.FilterMessage no longer throws for re-entrant implementations of IMessageFilter.PreFilterMessage

Details

Prior to the .NET Framework 4.6.1, calling [FilterMessage\(Message\)](#) with an [PreFilterMessage\(Message\)](#) which called

`System.Windows.Forms.Application.AddMessageFilter(IMessageFilter)` or `System.Windows.Forms.Application.RemoveMessageFilter(IMessageFilter)` (while also calling `DoEvents()`) would cause an `System.IndexOutOfRangeException`.

Beginning with applications targeting the .NET Framework 4.6.1, this exception is no longer thrown, and re-entrant filters as described above may be used.

Suggestion

Be aware that `FilterMessage(Message)` will no longer throw for the re-entrant `PreFilterMessage(Message)` behavior described above. This only affects applications targeting the .NET Framework 4.6.1. Apps targeting the .NET Framework 4.6.1 can opt out of this change (or apps targeting older Frameworks may opt in) by using the `DontSupportReentrantFilterMessage` compatibility switch.

Name	Value
Scope	Edge
Version	4.6.1
Type	Retargeting

Affected APIs

- `Application.FilterMessage(Message)`

Windows Presentation Foundation (WPF)

Calls to `System.Windows.Input.PenContext.Disable` on touch-enabled systems may throw an `ArgumentException`

Details

Under some circumstances, calls to the internal `System.Windows.Input.PenContext.Disable` method on touch-enabled systems may throw an unhandled `T:System.ArgumentException` because of reentrancy.

Suggestion

This issue has been addressed in the .NET Framework 4.7. To prevent the exception, upgrade to a version of the .NET Framework starting with the .NET Framework 4.7.

Name	Value
Scope	Edge
Version	4.6.1
Type	Retargeting

.NET Framework 4.6.2

ASP.NET

HttpRuntime.AppDomainAppPath Throws a NullReferenceException

Details

In the .NET Framework 4.6.2, the runtime throws a `T:System.NullReferenceException` when retrieving a `P:System.Web.HttpRuntime.AppDomainAppPath` value that includes null characters. In the .NET Framework 4.6.1 and earlier versions, the runtime throws an `T:System.ArgumentNullException`.

Suggestion

You can do either of the follow to respond to this change:

- Handle the `T:System.NullReferenceException` if you application is running on the .NET Framework 4.6.2.
- Upgrade to the .NET Framework 4.7, which restores the previous behavior and throws an `T:System.ArgumentNullException`.

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [HttpRuntime.AppDomainAppPath](#)

Core

AesCryptoServiceProvider decryptor provides a reusable transform

Details

Starting with apps that target the .NET Framework 4.6.2, the [AesCryptoServiceProvider](#) decryptor provides a reusable transform. After a call to [System.Security.Cryptography.CryptoAPITransform.TransformFinalBlock\(Byte\[\], Int32, Int32\)](#), the transform is reinitialized and can be reused. For apps that target earlier versions of the .NET Framework, attempting to reuse the decryptor by calling [System.Security.Cryptography.CryptoAPITransform.TransformBlock\(Byte\[\], Int32, Int32, Byte\[\], Int32\)](#) after a call to [System.Security.Cryptography.CryptoAPITransform.TransformFinalBlock\(Byte\[\], Int32, Int32\)](#) throws a [CryptographicException](#) or produces corrupted data.

Suggestion

The impact of this change should be minimal, since this is the expected behavior. Applications that depend on the previous behavior can opt out of it by adding the following configuration setting to the `<runtime>` section of the application's configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Security.Cryptography.AesCryptoServiceProvider.DontCorr
ectlyResetDecryptor=true"/>
</runtime>
```

In addition, applications that target a previous version of the .NET Framework but are running under a version of the .NET Framework starting with .NET Framework 4.6.2 can opt in to it by adding the following configuration setting to the `<runtime>` section of the application's configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Security.Cryptography.AesCryptoServiceProvider.DontCorruptDecryption=false"/>
</runtime>
```

Name	Value
Scope	Minor
Version	4.6.2
Type	Retargeting

Affected APIs

- [AesCryptoServiceProvider.CreateDecryptor\(\)](#)

Calls to ClaimsIdentity constructors

Details

Starting with the .NET Framework 4.6.2, there is a change in how [ClaimsIdentity](#) constructors with an [System.Security.Principal.IIdentity](#) parameter set the [System.Security.Claims.ClaimsIdentity.Actor](#) property. If the [System.Security.Principal.IIdentity](#) argument is a [ClaimsIdentity](#) object, and the [System.Security.Claims.ClaimsIdentity.Actor](#) property of that [ClaimsIdentity](#) object is not `null`, the [System.Security.Claims.ClaimsIdentity.Actor](#) property is attached by using the [Clone\(\)](#) method. In the Framework 4.6.1 and earlier versions, the [System.Security.Claims.ClaimsIdentity.Actor](#) property is attached as an existing reference. Because of this change, starting with the .NET Framework 4.6.2, the [System.Security.Claims.ClaimsIdentity.Actor](#) property of the new [ClaimsIdentity](#) object is not equal to the [System.Security.Claims.ClaimsIdentity.Actor](#) property of the constructor's [System.Security.Principal.IIdentity](#) argument. In the .NET Framework 4.6.1 and earlier versions, it is equal.

Suggestion

If this behavior is undesirable, you can restore the previous behavior by setting the [Switch.System.Security.ClaimsIdentity.SetActorAsReferenceWhenCopyingClaimsIdentity](#)

switch in your application configuration file to `true`. This requires that you add the following to the `<runtime>` section of your web.config file:

```
XML

<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.Security.ClaimsIdentity.SetActorAsReferenceWhenCopyingClaimsIdentity=true" />
  </runtime>
</configuration>
```

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- `ClaimsIdentity(IIdentity)`
- `ClaimsIdentity(IIdentity, IEnumerable<Claim>)`
- `ClaimsIdentity(IIdentity, IEnumerable<Claim>, String, String, String)`

Changes in path normalization

Details

Starting with apps that target the .NET Framework 4.6.2, the way in which the runtime normalizes paths has changed. Normalizing a path involves modifying the string that identifies a path or file so that it conforms to a valid path on the target operating system. Normalization typically involves:

- Canonicalizing component and directory separators.
- Applying the current directory to a relative path.
- Evaluating the relative directory `(.)` or the parent directory `(..)` in a path.
- Trimming specified characters. Starting with apps that target the .NET Framework 4.6.2, the following changes in path normalization are enabled by default:
 - The runtime defers to the operating system's `GetFullPathName` function to normalize paths.

- Normalization no longer involves trimming the end of directory segments (such as a space at the end of a directory name).
- Support for device path syntax in full trust, including `\.\.` and, for file I/O APIs in `mscorlib.dll`, `\?\.`.
- The runtime does not validate device syntax paths.
- The use of device syntax to access alternate data streams is supported. These changes improve performance while allowing methods to access previously inaccessible paths. Apps that target the .NET Framework 4.6.1 and earlier versions but are running under the .NET Framework 4.6.2 or later are unaffected by this change.

Suggestion

Apps that target the .NET Framework 4.6.2 or later can opt out of this change and use legacy normalization by adding the following to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.UseLegacyPathHandling=true" />
</runtime>
```

Apps that target the .NET Framework 4.6.1 or earlier but are running on the .NET Framework 4.6.2 or later can enable the changes to path normalization by adding the following line to the `<runtime>` section of the application .configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.UseLegacyPathHandling=false" />
</runtime>
```

Name	Value
Scope	Minor
Version	4.6.2
Type	Retargeting

CurrentCulture and CurrentUICulture flow across tasks

Details

Beginning in the .NET Framework 4.6, `System.Globalization.CultureInfo.CurrentCulture` and `System.Globalization.CultureInfo.CurrentUICulture` are stored in the thread's `System.Threading.ExecutionContext`, which flows across asynchronous operations. This means that changes to `System.Globalization.CultureInfo.CurrentCulture` or `System.Globalization.CultureInfo.CurrentUICulture` will be reflected in tasks which are later run asynchronously. This is different from the behavior of previous .NET Framework versions (which would reset `System.Globalization.CultureInfo.CurrentCulture` and `System.Globalization.CultureInfo.CurrentUICulture` in all asynchronous tasks).

Suggestion

Apps affected by this change may work around it by explicitly setting the desired `System.Globalization.CultureInfo.CurrentCulture` or `System.Globalization.CultureInfo.CurrentUICulture` as the first operation in an async Task. Alternatively, the old behavior (of not flowing `System.Globalization.CultureInfo.CurrentCulture/System.Globalization.CultureInfo.CurrentUICulture`) may be opted into by setting the following compatibility switch:

C#

```
ApplicationContext.SetSwitch("Switch.System.Globalization.NoAsyncCurrentCulture",  
    true);
```

This issue has been fixed by WPF in .NET Framework 4.6.2. It has also been fixed in .NET Frameworks 4.6, 4.6.1 through [KB 3139549](#). Applications targeting .NET Framework 4.6 or later will automatically get the right behavior in WPF applications - `System.Globalization.CultureInfo.CurrentCulture/System.Globalization.CultureInfo.CurrentUICulture` would be preserved across Dispatcher operations.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Affected APIs

- `CultureInfo.CurrentCulture`
- `Thread.CurrentCulture`
- `CultureInfo.CurrentUICulture`
- `Thread.CurrentUICulture`

ETW event names cannot differ only by a "Start" or "Stop" suffix

Details

In the .NET Framework 4.6 and 4.6.1, the runtime throws an [ArgumentException](#) when two Event Tracing for Windows (ETW) event names differ only by a "Start" or "Stop" suffix (as when one event is named `LogUser` and another is named `LogUserStart`). In this case, the runtime cannot construct the event source, which cannot emit any logging.

Suggestion

To prevent the exception, ensure that no two event names differ only by a "Start" or "Stop" suffix. This requirement is removed starting with the .NET Framework 4.6.2; the runtime can disambiguate event names that differ only by the "Start" and "Stop" suffix.

Name	Value
Scope	Edge
Version	4.6
Type	Retargeting

Long path support

Details

Starting with apps that target the .NET Framework 4.6.2, long paths (of up to 32K characters) are supported, and the 260-character (or `MAX_PATH`) limitation on path lengths has been removed. For apps that are recompiled to target the .NET Framework 4.6.2, code paths that previously threw a [System.IO.PathTooLongException](#) because a path exceeded 260 characters will now throw a [System.IO.PathTooLongException](#) only under the following conditions:

- The length of the path is greater than `.MaxValue` (32,767) characters.

- The operating system returns `COR_E_PATHTOOLONG` or its equivalent. For apps that target the .NET Framework 4.6.1 and earlier versions, the runtime automatically throws a [System.IO.PathTooLongException](#) whenever a path exceeds 260 characters.

Suggestion

For apps that target the .NET Framework 4.6.2, you can opt out of long path support if it is not desirable by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides value="Switch.System.IO.BlockLongPaths=true" />
</runtime>
```

For apps that target earlier versions of the .NET Framework but run on the .NET Framework 4.6.2 or later, you can opt in to long path support by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides value="Switch.System.IO.BlockLongPaths=false"
/>
</runtime>
```

Name	Value
Scope	Minor
Version	4.6.2
Type	Retargeting

Path colon checks are stricter

Details

In .NET Framework 4.6.2, a number of changes were made to support previously unsupported paths (both in length and format). Checks for proper drive separator

(colon) syntax were made more correct, which had the side effect of blocking some URI paths in a few select Path APIs where they were previously tolerated.

Suggestion

If passing a URI to affected APIs, modify the string to be a legal path first.

- Remove the scheme from URLs manually (for example, remove `file://` from URLs).
- Pass the URI to the [Uri](#) class and use [LocalPath](#).

Alternatively, you can opt out of the new path normalization by setting the `Switch.System.IO.UseLegacyPathHandling` ApplicationContext switch to `true`.

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [Path.GetDirectoryName\(String\)](#)
- [Path.GetPathRoot\(String\)](#)

Security

RSACng now correctly loads RSA keys of non-standard key size

Details

In .NET Framework versions prior to 4.6.2, customers with non-standard key sizes for RSA certificates are unable to access those keys via the [System.Security.Cryptography.X509Certificates.RSACertificateExtensions.GetRSAPublicKey\(X509Certificate2\)](#) and [System.Security.Cryptography.X509Certificates.RSACertificateExtensions.GetRSAPrivateKey\(X509Certificate2\)](#) extension methods. A [System.Security.Cryptography.CryptographicException](#) with the message "The requested

"key size is not supported" is thrown. In .NET Framework 4.6.2 this issue has been fixed. Similarly, [ImportParameters\(RSAParameters\)](#) and [ImportParameters\(RSAParameters\)](#) now work with non-standard key sizes without throwing a [System.Security.Cryptography.CryptographicException](#).

Suggestion

If there is any exception handling logic that relies on the previous behavior where a [System.Security.Cryptography.CryptographicException](#) is thrown when non-standard key sizes are used, consider removing the logic.

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [RSA.ImportParameters\(RSAParameters\)](#)
- [RSACng.ImportParameters\(RSAParameters\)](#)
- [RSACertificateExtensions.GetRSAPrivateKey\(X509Certificate2\)](#)
- [RSACertificateExtensions.GetRSAPublicKey\(X509Certificate2\)](#)

SignedXml.GetPublicKey returns RSACng on net462 (or lightup) without retargeting change

Details

Starting with the .NET Framework 4.6.2, the concrete type of the object returned by the [SignedXml.GetPublicKey](#) method changed (without a quirk) from a [CryptoServiceProvider](#) implementation to a Cng implementation. This is because the implementation changed from using `certificate.PublicKey.Key` to using the internal `certificate.GetAnyPublicKey` which forwards to [RSACertificateExtensions.GetRSAPublicKey](#).

Suggestion

Starting with apps running on the .NET Framework 4.7.1, you can use the `CryptoServiceProvider` implementation used by default in the .NET Framework 4.6.1 and earlier versions by adding the following configuration switch to the `runtime` section of your app config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Security.Cryptography.Xml.SignedXmlUseLegacyCertificate  
PrivateKey=true" />
```

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- `SignedXml.CheckSignatureReturningKey(AsymmetricAlgorithm)`

Windows Communication Foundation (WCF)

Deadlock may result when using Reentrant services

Details

A deadlock may result in a Reentrant service, which restricts instances of the service to one thread of execution at a time. Services prone to encounter this problem will have the following `ServiceBehaviorAttribute` in their code:

C#

```
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Reentrant)]
```

Suggestion

To address this issue, you can do the following:

- Set the service's concurrency mode to `ConcurrencyMode.Single` or `ConcurrencyMode.Multiple`. For example:

C#

```
[ServiceBehavior(ConcurrencyMode = ConcurrencyMode.Reentrant)]
```

- Install the latest update to the .NET Framework 4.6.2, or upgrade to a later version of the .NET Framework. This disables the flow of the `ExecutionContext` in `OperationContext.Current`. This behavior is configurable; it is equivalent to adding the following app setting to your configuration file:

XML

```
<appSettings>
  <add key="Switch.System.ServiceModel.DisableOperationContextAsyncFlow"
       value="true" />
</appSettings>
```

The value of `Switch.System.ServiceModel.DisableOperationContextAsyncFlow` should never be set to `false` for Reentrant services.

Name	Value
Scope	Minor
Version	4.6.2
Type	Retargeting

Affected APIs

- `System.ServiceModel.ServiceBehaviorAttribute`
- `ConcurrencyMode.Reentrant`

OperationContext.Current may return null when called in a using clause

Details

`OperationContext.Current` may return `null` and a `NullReferenceException` may result if all of the following conditions are true:

- You retrieve the value of the `OperationContext.Current` property in a method that returns a `Task` or `Task<TResult>`.
- You instantiate the `OperationContextScope` object in a `using` clause.
- You retrieve the value of the `OperationContext.Current` property within the `using` statement. For example:

C#

```
using (new OperationContextScope(OperationContext.Current))
{
    // OperationContext.Current is null.
    OperationContext context = OperationContext.Current;

    // ...
}
```

Suggestion

To address this issue, you can do the following:

- Modify your code as follows to instantiate a new non-`null` `Current` object:

C#

```
OperationContext ocx = OperationContext.Current;
using (new OperationContextScope(OperationContext.Current))
{
    OperationContext.Current = new OperationContext(ocx.Channel);

    // ...
}
```

- Install the latest update to the .NET Framework 4.6.2, or upgrade to a later version of the .NET Framework. This disables the flow of the `ExecutionContext` in `OperationContext.Current` and restores the behavior of WCF applications in the .NET Framework 4.6.1 and earlier versions. This behavior is configurable; it is equivalent to adding the following app setting to your configuration file:

XML

```
<appSettings>
    <add
        key="Switch.System.ServiceModel.DisableOperationContextAsyncFlow"
        value="true" />
</appSettings>
```

If this change is undesirable and your application depends on execution context flowing between operation contexts, you can enable its flow as follows:

XML

```
<appSettings>
  <add
    key="Switch.System.ServiceModel.DisableOperationContextAsyncFlow"
    value="false" />
</appSettings>
```

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [OperationContext.Current](#)

WCF transport security supports certificates stored using CNG

Details

Starting with apps that target the .NET Framework 4.6.2, WCF transport security supports certificates stored using the Windows Cryptography Library (CNG). This support is limited to certificates with a public key that has an exponent no more than 32 bits in length. When an application targets the .NET Framework 4.6.2, this feature is on by default. In earlier versions of the .NET Framework, the attempt to use X509 certificates with a CSG key storage provider throws an exception.

Suggestion

Apps that target the .NET Framework 4.6.1 and earlier but are running on the .NET Framework 4.6.2 can enable support for CNG certificates by adding the following line to the `<runtime>` section of the app.config or web.config file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IdentityModel.DisableCngCertificates=false" />
</runtime>
```

This can also be done programmatically with the following code:

C#

```
private const string DisableCngCertificates =
@"Switch.System.IdentityModel.DisableCngCertificate";
AppContext.SetSwitch(disableCngCertificates, false);
```

VB

```
Const DisableCngCertificates As String =
"Switch.System.IdentityModel.DisableCngCertificates"
AppContext.SetSwitch(disableCngCertificates, False)
```

Note that, because of this change, any exception handling code that depends on the attempt to initiate secure communication with a CNG certificate to fail will no longer execute.

Name	Value
Scope	Minor
Version	4.6.2
Type	Retargeting

Windows Forms

Incorrect implementation of MemberDescriptor.Equals

Details

The original implementation of the [MemberDescriptor.Equals](#) method compares two different string properties from the objects being compared: the category name and the description string. The fix is to compare the [Category](#) of the first object to the [Category](#) of the second one, and the [Description](#) of the first to the [Description](#) of the second.

Suggestion

If your application depends on [MemberDescriptor.Equals](#) sometimes returning `false` when descriptors are equivalent, and you are targeting the .NET Framework 4.6.2 or later, you have several options:

- Make code changes to compare the [Category](#) and [Description](#) fields manually in addition to calling the [MemberDescriptor.Equals](#) method.
- Opt out of this change by adding the following value to the app.config file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.MemberDescriptorEqualsReturnsFalseIfEquivalent=true" />
</runtime>
```

If your application targets .NET Framework 4.6.1 or earlier and is running on the .NET Framework 4.6.2 or later and you want this change enabled, you can set the compatibility switch to `false` by adding the following value to the app.config file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.MemberDescriptorEqualsReturnsFalseIfEquivalent=false"
  />
</runtime>
```

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [MemberDescriptor.Equals\(Object\)](#)

Windows Presentation Foundation (WPF)

CurrentCulture is not preserved across WPF Dispatcher operations

Details

Beginning in the .NET Framework 4.6, changes to [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) made within a [System.Windows.Threading.Dispatcher](#) will be lost at the end of that dispatcher operation. Similarly, changes to [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) made outside of a Dispatcher operation may not be reflected when that operation executes. Practically speaking, this means that [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) changes may not flow between WPF UI callbacks and other code in a WPF application. This is due to a change in [System.Threading.ExecutionContext](#) that causes [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) to be stored in the execution context beginning with apps targeting the .NET Framework 4.6. WPF dispatcher operations store the execution context used to begin the operation and restore the previous context when the operation is completed. Because [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) are now part of that context, changes to them within a dispatcher operation are not persisted outside of the operation.

Suggestion

Apps affected by this change may work around it by storing the desired [System.Globalization.CultureInfo.CurrentCulture](#) or [System.Globalization.CultureInfo.CurrentUICulture](#) in a field and checking in all Dispatcher operation bodies (including UI event callback handlers) that the correct [System.Globalization.CultureInfo.CurrentCulture](#) and [System.Globalization.CultureInfo.CurrentUICulture](#) are set. Alternatively, because the [ExecutionContext](#) change underlying this WPF change only affects apps targeting the .NET Framework 4.6 or newer, this break can be avoided by targeting the .NET Framework 4.5.2. Apps that target .NET Framework 4.6 or later can also work around this by setting the following compatibility switch:

C#

```
ApplicationContext.SetSwitch("Switch.System.Globalization.NoAsyncCurrentCulture",  
true);
```

This issue has been fixed by WPF in .NET Framework 4.6.2. It has also been fixed in .NET Frameworks 4.6, 4.6.1 through [KB 3139549](#). Applications targeting .NET Framework 4.6 or later will automatically get the right behavior in WPF applications - [System.Globalization.CultureInfo.CurrentCulture/System.Globalization.CultureInfo.CurrentUICulture](#)) would be preserved across Dispatcher operations.

Name	Value
Scope	Minor
Version	4.6
Type	Retargeting

Retargeting changes for migration to .NET Framework 4.7.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework [4.7](#), [4.7.1](#), and [4.7.2](#).

.NET Framework 4.7

ASP.NET

HttpRuntime.AppDomainAppPath Throws a NullReferenceException

Details

In the .NET Framework 4.6.2, the runtime throws a `T:System.NullReferenceException` when retrieving a `P:System.Web.HttpRuntime.AppDomainAppPath` value that includes null characters. In the .NET Framework 4.6.1 and earlier versions, the runtime throws an `T:System.ArgumentNullException`.

Suggestion

You can do either of the follow to respond to this change:

- Handle the `T:System.NullReferenceException` if your application is running on the .NET Framework 4.6.2.
- Upgrade to the .NET Framework 4.7, which restores the previous behavior and throws an `T:System.ArgumentNullException`.

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [HttpRuntime.AppDomainAppPath](#)

Throttle concurrent requests per session

Details

In the .NET Framework 4.6.2 and earlier, ASP.NET executes requests with the same Sessionid sequentially, and ASP.NET always issues the Sessionid through cookies by default. If a page takes a long time to respond, it will significantly degrade server performance just by pressing **F5** on the browser. In the fix, we added a counter to track the queued requests and terminate the requests when they exceed a specified limit. The default value is 50. If the limit is reached, a warning will be logged in the event log, and an HTTP 500 response may be recorded in the IIS log.

Suggestion

To restore the old behavior, you can add the following setting to your web.config file to opt out of the new behavior.

XML

```
<appSettings>
  <add key="aspnet:RequestQueueLimitPerSession" value="2147483647"/>
</appSettings>
```

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Networking

Default value of ServicePointManager.SecurityProtocol is SecurityProtocolType.System.Default

Details

Starting with apps that target the .NET Framework 4.7, the default value of the [ServicePointManager.SecurityProtocol](#) property is [SecurityProtocolType.SystemDefault](#). This change allows .NET Framework networking APIs based on SslStream (such as FTP, HTTPS, and SMTP) to inherit the default security protocols from the operating system instead of using hard-coded values defined by the .NET Framework. The default varies by operating system and any custom configuration performed by the system administrator. For information on the default SChannel protocol in each version of the Windows operating system, see [Protocols in TLS/SSL \(Schannel SSP\)](#).

For applications that target an earlier version of the .NET Framework, the default value of the [ServicePointManager.SecurityProtocol](#) property depends on the version of the .NET Framework targeted. See the [Networking section of Retargeting Changes for Migration from .NET Framework 4.5.2 to 4.6](#) for more information.

Suggestion

This change affects applications that target the .NET Framework 4.7 or later versions. If you prefer to use a defined protocol rather than relying on the system default, you can explicitly set the value of the [ServicePointManager.SecurityProtocol](#) property. If this change is undesirable, you can opt out of it by adding a configuration setting to the [`<runtime>`](#) section of your application configuration file. The following example shows both the [`<runtime>`](#) section and the

`Switch.System.Net.DontEnableSystemDefaultTlsVersions` opt-out switch:

```
XML

<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Net.DontEnableSystemDefaultTlsVersions=true" />
</runtime>
```

Name	Value
Scope	Minor
Version	4.7
Type	Retargeting

Affected APIs

- [ServicePointManager.SecurityProtocol](#)

SslStream supports TLS Alerts

Details

After a failed TLS handshake, an [System.IO.IOException](#) with an inner [System.ComponentModel.Win32Exception](#) exception will be thrown by the first I/O Read/Write operation. The [System.ComponentModel.Win32Exception.NativeErrorCode](#) code for the [System.ComponentModel.Win32Exception](#) can be mapped to the TLS Alert from the remote party using the [Schannel error codes for TLS and SSL alerts](#). For more information, see [RFC 2246: Section 7.2.2 Error alerts](#).

The behavior in .NET Framework 4.6.2 and earlier is that the transport channel (usually TCP connection) will timeout during either Write or Read if the other party failed the handshake and immediately afterwards rejected the connection.

Suggestion

Applications calling network I/O APIs such as [Read\(Byte\[\], Int32, Int32\)](#)/[Write\(Byte\[\], Int32, Int32\)](#) should handle [IOException](#) or [System.TimeoutException](#).

The TLS Alerts feature is enabled by default starting with .NET Framework 4.7.

Applications targeting versions of the .NET Framework from 4.0 through 4.6.2 running on a .NET Framework 4.7 or higher system will have the feature disabled to preserve compatibility.

The following configuration API is available to enable or disable the feature for .NET Framework 4.6 and later applications running on .NET Framework 4.7 or later.

- Programmatically: Must be the very first thing the application does since ServicePointManager will initialize only once:

C#

```
ApplicationContext.SetSwitch("TestSwitch.Local ApplicationContext.DisableCaching",  
    true);  
  
    // Set to 'false' to enable the feature in .NET Framework 4.6 - 4.6.2.  
    ApplicationContext.SetSwitch("Switch.System.Net.DontEnableTlsAlerts", true);
```

- AppConfig:

XML

```
<runtime>  
    <AppContextSwitchOverrides  
        value="Switch.System.Net.DontEnableTlsAlerts=true" />
```

```
<!-- Set to 'false' to enable the feature in .NET Framework 4.6 -  
4.6.2. -->  
</runtime>
```

- Registry key (machine global): Set the Value to `false` to enable the feature in .NET Framework 4.6 - 4.6.2.

ini

Key:
HKLM\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\ApplicationContext\Switch.System.Net.DontEnableTlsAlerts
- Type: String
- Value: "true"

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Affected APIs

- [System.Net.Security.SslStream](#)
- [System.Net.WebRequest](#)
- [System.Net.HttpWebRequest](#)
- [System.Net.FtpWebRequest](#)
- [System.Net.Mail.SmtpClient](#)
- [System.Net.Http](#)

Security

CspParameters.ParentWindowHandle now expects HWND value

Details

The `ParentWindowHandle` value, introduced in .NET Framework 2.0, allows an application to register a parent window handle value such that any UI required to access the key (such as a PIN prompt or consent dialog) opens as a modal child to the specified

Starting with apps that target the .NET Framework 4.7, a Windows Forms application can set the `ParentWindowHandle` property with code like the following:

```
C#
```

```
cspParameters.ParentWindowHandle = form.Handle;
```

In previous versions of the .NET Framework, the value was expected to be an `System.IntPtr` representing a location in memory where the `HWND` value resided. Setting the property to `form.Handle` on Windows 7 and earlier versions had no effect, but on Windows 8 and later versions, it results in a `"System.Security.Cryptography.CryptographicException": The parameter is incorrect."`

Suggestion

Applications targeting .NET Framework 4.7 or higher wishing to register a parent window relationship are encouraged to use the simplified form:

```
C#
```

```
cspParameters.ParentWindowHandle = form.Handle;
```

Users who had identified that the correct value to pass was the address of a memory location which held the value `form.Handle` can opt out of the behavior change by setting the `AppContext` switch

`Switch.System.Security.Cryptography.DoNotAddrOfCspParentWindowHandle` to `true`:

- By programmatically setting compat switches on the `AppContext`, as explained [here](#).
- By adding the following line to the `<runtime>` section of the `app.config` file:

```
XML
```

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Security.Cryptography.DoNotAddrOfCspParentWindowHandle=
      true"/>
</runtime>
```

Conversely, users who wish to opt in to the new behavior on the .NET Framework 4.7 runtime when the application loads under older .NET Framework versions can set the `AppContext` switch to `false`.

Name	Value
Scope	Minor
Version	4.7
Type	Retargeting

Affected APIs

- [CspParameters.ParentWindowHandle](#)

SslStream supports TLS Alerts

Details

After a failed TLS handshake, an [System.IO.IOException](#) with an inner [System.ComponentModel.Win32Exception](#) exception will be thrown by the first I/O Read/Write operation. The [System.ComponentModel.Win32Exception.NativeErrorCode](#) code for the [System.ComponentModel.Win32Exception](#) can be mapped to the TLS Alert from the remote party using the [Schannel error codes for TLS and SSL alerts](#). For more information, see [RFC 2246: Section 7.2.2 Error alerts](#).

The behavior in .NET Framework 4.6.2 and earlier is that the transport channel (usually TCP connection) will timeout during either Write or Read if the other party failed the handshake and immediately afterwards rejected the connection.

Suggestion

Applications calling network I/O APIs such as [Read\(Byte\[\], Int32, Int32\)/Write\(Byte\[\], Int32, Int32\)](#) should handle [IOException](#) or [System.TimeoutException](#).

The TLS Alerts feature is enabled by default starting with .NET Framework 4.7.

Applications targeting versions of the .NET Framework from 4.0 through 4.6.2 running on a .NET Framework 4.7 or higher system will have the feature disabled to preserve compatibility.

The following configuration API is available to enable or disable the feature for .NET Framework 4.6 and later applications running on .NET Framework 4.7 or later.

- Programmatically: Must be the very first thing the application does since ServicePointManager will initialize only once:

C#

```

    ApplicationContext.SetSwitch("TestSwitch.LocalApplicationContext.DisableCaching",
        true);

    // Set to 'false' to enable the feature in .NET Framework 4.6 - 4.6.2.
    ApplicationContext.SetSwitch("Switch.System.Net.DontEnableTlsAlerts", true);

```

- AppConfig:

XML

```

<runtime>
    < ApplicationContextSwitchOverrides
        value="Switch.System.Net.DontEnableTlsAlerts=true" />
        <!-- Set to 'false' to enable the feature in .NET Framework 4.6 -
        4.6.2. -->
    </runtime>

```

- Registry key (machine global): Set the Value to `false` to enable the feature in .NET Framework 4.6 - 4.6.2.

ini

Key:
HKLM\SOFTWARE\Wow6432Node\Microsoft\.NETFramework\ApplicationContext\Switch.System.Net.DontEnableTlsAlerts
- Type: String
- Value: "true"

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Affected APIs

- [System.Net.Security.SslStream](#)
- [System.Net.WebRequest](#)
- [System.Net.HttpWebRequest](#)
- [System.Net.FtpWebRequest](#)
- [System.Net.Mail.SmtpClient](#)
- [System.Net.Http](#)

Windows Communication Foundation (WCF)

Serialization of control characters with DataContractJsonSerializer is now compatible with ECMAScript V6 and V8

Details

In .NET Framework 4.6.2 and earlier versions, the [System.Runtime.Serialization.Json.DataContractJsonSerializer](#) did not serialize some special control characters, such as \b, \f, and \t, in a way that was compatible with the ECMAScript V6 and V8 standards. Starting with .NET Framework 4.7, serialization of these control characters is compatible with ECMAScript V6 and V8.

Suggestion

For apps that target the .NET Framework 4.7, this feature is enabled by default. If this behavior is not desirable, you can opt out of this feature by adding the following line to the `<runtime>` section of the app.config or web.config file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Runtime.Serialization.DoNotUseECMAScriptV6EscapeControl
Character=false" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Affected APIs

- [DataContractJsonSerializer.WriteObject\(Stream, Object\)](#)
- [DataContractJsonSerializer.WriteObject\(XmlDictionaryWriter, Object\)](#)
- [DataContractJsonSerializer.WriteObject\(XmlWriter, Object\)](#)

WCF message security now is able to use TLS1.1 and TLS1.2

Details

Starting in the .NET Framework 4.7, customers can configure either TLS1.1 or TLS1.2 in WCF message security in addition to SSL3.0 and TLS1.0 through application configuration settings.

Suggestion

In the .NET Framework 4.7, support for TLS1.1 and TLS1.2 in WCF message security is disabled by default. You can enable it by adding the following line to the `<runtime>` section of the app.config or web.config file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.ServiceModel.DisableUsingServicePointManagerSecurityProtocols=false;Switch.System.Net.DontEnableSchUseStrongCrypto=false" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Windows Presentation Foundation (WPF)

Calls to System.Windows.Input.PenContext.Disable on touch-enabled systems may throw an ArgumentException

Details

Under some circumstances, calls to the internal `System.Windows.Input.PenContext.Disable` method on touch-enabled systems may throw an unhandled `T:System.ArgumentException` because of reentrancy.

Suggestion

This issue has been addressed in the .NET Framework 4.7. To prevent the exception, upgrade to a version of the .NET Framework starting with the .NET Framework 4.7.

Name	Value
Scope	Edge
Version	4.6.1
Type	Retargeting

NullReferenceException in exception handling code from `ImageSourceConverter.ConvertFrom`

Details

An error in the exception handling code for `ConvertFrom(ITypeDescriptorContext, CultureInfo, Object)` caused an incorrect `System.NullReferenceException` to be thrown instead of the intended exception (`System.IO.DirectoryNotFoundException` or `System.IO.FileNotFoundException`). This change corrects that error so that the method now throws the right exception.

By default all applications targeting .NET Framework 4.6.2 and earlier continue to throw `System.NullReferenceException` for compatibility. Developers targeting .NET Framework 4.7 and above should see the right exceptions.

Suggestion

Developers who wish to revert to getting `System.NullReferenceException` when targeting .NET Framework 4.7 or later can add/merge the following to their application's App.config file:

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Media.ImageSourceConverter.OverrideExceptionWithNullReferenceException=true"/>
</runtime>
</configuration>
```

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Affected APIs

- [ImageSourceConverter.ConvertFrom\(ITypeDescriptorContext, CultureInfo, Object\)](#)

WPF Grid allocation of space to star-columns

Details

Starting with the .NET Framework 4.7, WPF replaces the algorithm that [Grid](#) uses to allocate space to *-columns. This will change the actual width assigned to *-columns in a number of cases:

- When one or more *-columns also have a minimum or maximum width that overrides the proportional allocation for that column. (The minimum width can derive from an explicit MinWidth declaration, or from an implicit minimum obtained from the column's content. The maximum width can only be defined explicitly, from a MaxWidth declaration.)
- When one or more *-columns declare an extremely large *-weight, greater than 10^{298} .
- When the *-weights are sufficiently different to encounter floating-point instability (overflow, underflow, loss of precision).
- When layout rounding is enabled, and the effective display DPI is sufficiently high. In the first two cases, the widths produced by the new algorithm can be significantly different from those produced by the old algorithm; in the last case, the difference will be at most one or two pixels.

The new algorithm fixes several bugs present in the old algorithm:

- Total allocation to columns can exceed the Grid's width. This can occur when allocating space to a column whose proportional share is less than its minimum size. The algorithm allocates the minimum size, which decreases the space

available to other columns. If there are no *-columns left to allocate, the total allocation will be too large.

- Total allocation can fall short of the Grid's width. This is the dual problem to #1, arising when allocating to a column whose proportional share is greater than its maximum size, with no *-columns left to take up the slack.
 - Two *-columns can receive allocations not proportional to their *-weights. This is a milder version of #1/#2, arising when allocating to *-columns A, B, and C (in that order), where B's proportional share violates its min (or max) constraint. As above, this changes the space available to column C, who gets less (or more) proportional allocation than A did,
 - Columns with extremely large weights ($> 10^{298}$) are all treated as if they had weight 10^{298} . Proportional differences between them (and between columns with slightly smaller weights) are not honored.
 - Columns with infinite weights are not handled correctly. [Actually you can't set a weight to Infinity, but this is an artificial restriction. The allocation code was trying to handle it, but doing a bad job.]
 - Several minor problems while avoiding overflow, underflow, loss of precision and similar floating-point issues.
 - Adjustments for layout rounding are incorrect at sufficiently high DPI. The new algorithm produces results that meet the following criteria:
 - A. The actual width assigned to a *-column is never less than its minimum width nor greater than its maximum width.
 - B. Each *-column that is not assigned its minimum or maximum width is assigned a width proportional to its -weight*. To be precise, if two columns are declared with width x and y respectively, and if neither column receives its minimum or maximum width, the actual widths v and w assigned to the columns are in the same proportion: $v / w == x / y$.
 - C. The total width allocated to "proportional" *-columns is equal to the space available after allocating to the constrained columns (fixed, auto, and *-columns that are allocated their min or max width). This might be zero, for instance if the sum of the minimum widths exceeds the Grid's available width.
 - D. All these statements are to be interpreted with respect to the "ideal" layout. When layout rounding is in effect, the actual widths can differ from the ideal widths by as much as one pixel.
- The old algorithm honored (A) but failed to honor the other criteria in the cases outlined above.

Everything said about columns and widths in this article applies as well to rows and heights.

Suggestion

By default, apps that target versions of the .NET Framework starting with the .NET Framework 4.7 will see the new algorithm, while apps that target the .NET Framework 4.6.2 or earlier versions will see the old algorithm.

To override the default, use the following configuration setting:

```
XML

<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.Grid.StarDefinitionsCanExceedAvailable
Space=true" />
</runtime>
```

The value `true` selects the old algorithm, `false` selects the new algorithm.

Name	Value
Scope	Minor
Version	4.7
Type	Retargeting

WPF Pointer-Based Touch Stack

Details

This change adds the ability to enable an optional WM_POINTER based WPF touch/stylus stack. Developers that do not explicitly enable this should see no change in WPF touch/stylus behavior. Current Known Issues With optional WM_POINTER based touch/stylus stack:

- No support for real-time inking.
- While inking and StylusPlugins will still work, they will be processed on the UI Thread which can lead to poor performance.
- Behavioral changes due to changes in promotion from touch/stylus events to mouse events

- Manipulation may behave differently
- Drag/Drop will not show appropriate feedback for touch input
- This does not affect stylus input
- Drag/Drop can no longer be initiated on touch/stylus events
- This can potentially cause the application to stop responding until mouse input is detected.
- Instead, developers should initiate drag and drop from mouse events.

Suggestion

Developers who wish to enable this stack can add/merge the following to their application's App.config file:

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Input.Stylus.EnablePointerSupport=true"/>
</runtime>
</configuration>
```

Removing this or setting the value to false will turn this optional stack off. Note that this stack is available only on Windows 10 Creators Update and above.

Name	Value
Scope	Edge
Version	4.7
Type	Retargeting

Windows Workflow Foundation (WF)

Workflow checksums changed from MD5 to SHA1

Details

To support debugging with Visual Studio, the Workflow runtime generates a checksum for a workflow instance using a hashing algorithm. In the .NET Framework 4.6.2 and earlier versions, workflow checksum hashing used the MD5 algorithm, which caused

issues on FIPS-enabled systems. Starting with the .NET Framework 4.7, the algorithm is SHA1. If your code has persisted these checksums, they will be incompatible.

Suggestion

If your code is unable to load workflow instances due to a checksum failure, try setting the `AppContext` switch "Switch.System.Activities.UseMD5ForWFDebugger" to true. In code:

C#

```
System.AppContext.SetSwitch("Switch.System.Activities.UseMD5ForWFDebugger",  
    true);
```

Or in configuration:

XML

```
<configuration>  
  <runtime>  
    <AppContextSwitchOverrides  
      value="Switch.System.Activities.UseMD5ForWFDebugger=true" />  
  </runtime>  
</configuration>
```

Name	Value
Scope	Minor
Version	4.7
Type	Retargeting

.NET Framework 4.7.1

ASP.NET

ASP.NET Accessibility Improvements in .NET Framework 4.7.1

Details

Starting with the .NET Framework 4.7.1, ASP.NET has improved how ASP.NET Web Controls work with accessibility technology in Visual Studio to better support ASP.NET customers. These include the following changes:

- Changes to implement missing UI accessibility patterns in controls, like the Add Field dialog in the Details View wizard, or the Configure ListView dialog of the ListView wizard.
- Changes to improve the display in High Contrast mode, like the Data Pager Fields Editor.
- Changes to improve the keyboard navigation experiences for controls, like the Fields dialog in the Edit Pager Fields wizard of the DataPager control, the Configure ObjectContext dialog, or the Configure Data Selection dialog of the Configure Data Source wizard.

Suggestion

How to opt in or out of these changes In order for the Visual Studio Designer to benefit from these changes, it must run on the .NET Framework 4.7.1 or later. The web application can benefit from these changes in either of the following ways:

- Install Visual Studio 2017 15.3 or later, which supports the new accessibility features with the following AppContext Switch by default.
- Opt out of the legacy accessibility behaviors by adding the `Switch.UseLegacyAccessibilityFeatures` AppContext switch to the `<runtime>` section in the devenv.exe.config file and setting it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<runtime>
...
<!-- AppContextSwitchOverrides value attribute is in the form of
'key1=true/false;key2=true/false' -->
<AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false" />
...
</runtime>
</configuration>
```

Applications that target the .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this AppContext switch to `true`.

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Core

SerialPort background thread exceptions

Details

Background threads created with [SerialPort](#) streams no longer terminate the process when OS exceptions are thrown.

In applications that target the .NET Framework 4.7 and earlier versions, a process is terminated when an operating system exception is thrown on a background thread created with a [SerialPort](#) stream.

In applications that target the .NET Framework 4.7.1 or a later version, background threads wait for OS events related to the active serial port and could crash in some cases, such as sudden removal of the serial port.

Suggestion

For apps that target the .NET Framework 4.7.1, you can opt out of the exception handling if it is not desirable by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.Ports.DoNotCatchSerialStreamThreadExceptions=true"
  />
</runtime>
```

For apps that target earlier versions of the .NET Framework but run on the .NET Framework 4.7.1 or later, you can opt in to the exception handling by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.Ports.DoNotCatchSerialStreamThreadExceptions=false"
  />
</runtime>
```

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Affected APIs

- [System.IO.Ports.SerialPort](#)

ServiceBase doesn't propagate OnStart exceptions

Details

In the .NET Framework 4.7 and earlier versions, exceptions thrown on service startup are not propagated to the caller of [ServiceBase.Run](#).

Starting with applications that target the .NET Framework 4.7.1, the runtime propagates exceptions to [ServiceBase.Run](#) for services that fail to start.

Suggestion

On service start, if there is an exception, that exception will be propagated. This should help diagnose cases where services fail to start.

If this behavior is undesirable, you can opt out of it by adding the following

`AppContextSwitchOverrides` element to the `runtime` section of your application configuration file:

XML

```
<AppContextSwitchOverrides
  value="Switch.System.ServiceProcess.DontThrowExceptionsOnStart=true" />
```

If your application targets an earlier version than 4.7.1 but you want to have this behavior, add the following `ApplicationContextSwitchOverrides` element to the `runtime` section of your application configuration file:

XML

```
<ApplicationContextSwitchOverrides  
value="Switch.System.ServiceProcess.DontThrowExceptionsOnStart=false" />
```

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Affected APIs

- [ServiceBase.Run\(ServiceBase\)](#)
- [ServiceBase.Run\(ServiceBase\[\]\)](#)

Security

Default SignedXML and SignedXMS algorithms changed to SHA256

Details

In the .NET Framework 4.7 and earlier, SignedXML and SignedCMS default to SHA1 for some operations. Starting with the .NET Framework 4.7.1, SHA256 is enabled by default for these operations. This change is necessary because SHA1 is no longer considered to be secure.

Suggestion

There are two new context switch values to control whether SHA1 (insecure) or SHA256 is used by default:

- [Switch.System.Security.Cryptography.Xml.UseInsecureHashAlgorithms](#)

- `Switch.System.Security.Cryptography.Pkcs.UseInsecureHashAlgorithms` For applications that target the .NET Framework 4.7.1 and later versions, if the use of SHA256 is undesirable, you can restore the default to SHA1 by adding the following configuration switch to the `runtime` section of your app config file:

XML

```
<AppContextSwitchOverrides
  value="Switch.System.Security.Cryptography.Xml.UseInsecureHashAlgorithms=true
  Switch.System.Security.Cryptography.Pkcs.UseInsecureHashAlgorithms=true"
/>
```

For applications that target the .NET Framework 4.7 and earlier versions, you can opt into this change by adding the following configuration switch to the `runtime` section of your app config file:

XML

```
<AppContextSwitchOverrides
  value="Switch.System.Security.Cryptography.Xml.UseInsecureHashAlgorithms=false
  Switch.System.Security.Cryptography.Pkcs.UseInsecureHashAlgorithms=false"
/>
```

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Affected APIs

- `System.Security.Cryptography.Pkcs.CmsSigner`
- `System.Security.Cryptography.Xml.SignedXml`
- `System.Security.Cryptography.Xml.Reference`

SignedXml.GetPublicKey returns RSACng on net462 (or lightup) without retargeting change

Details

Starting with the .NET Framework 4.6.2, the concrete type of the object returned by the [SignedXml.GetPublicKey](#) method changed (without a quirk) from a [CryptoServiceProvider](#) implementation to a Cng implementation. This is because the implementation changed from using `certificate.PublicKey.Key` to using the internal `certificate.GetAnyPublicKey` which forwards to [RSACertificateExtensions.GetRSAPublicKey](#).

Suggestion

Starting with apps running on the .NET Framework 4.7.1, you can use the [CryptoServiceProvider](#) implementation used by default in the .NET Framework 4.6.1 and earlier versions by adding the following configuration switch to the [runtime](#) section of your app config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Security.Cryptography.Xml.SignedXmlUseLegacyCertificate  
PrivateKey=true" />
```

Name	Value
Scope	Edge
Version	4.6.2
Type	Retargeting

Affected APIs

- [SignedXml.CheckSignatureReturningKey\(AsymmetricAlgorithm\)](#)

Windows Communication Foundation (WCF)

Improved accessibility for some .NET SDK tools

Details

In the .NET Framework SDK 4.7.1, the SvcConfigEditor.exe and SvcTraceViewer.exe tools have been improved by fixing varied accessibility issues. Most of these were small issues like a name not being defined or certain UI automation patterns not being implemented.

correctly. While many users wouldn't be aware of these incorrect values, customers who use assistive technologies like screen readers will find these SDK tools more accessible. Certainly, these fixes change some previous behaviors, like keyboard focus order. In order to get all the accessibility fixes in these tools, you can add the following to your app.config file:

```
XML
<runtime>
  <AppContextSwitchOverrides
    value="Switch.UseLegacyAccessibilityFeatures=false"/>
</runtime>
```

Name	Value
Scope	Edge
Version	4.7.1
Type	Retargeting

Windows Forms

Accessibility improvements in Windows Forms controls

Details

Windows Forms is improving how it works with accessibility technologies to better support Windows Forms customers. These include the following changes starting with the .NET Framework 4.7.1:

- Changes to improve display during High Contrast mode.
- Changes to improve the property browser experience. Property browser improvements include:
 - Better keyboard navigation through the various drop-down selection windows.
 - Reduced unnecessary tab stops.
 - Better reporting of control types.
 - Improved narrator behavior.
 - Changes to implement missing UI accessibility patterns in controls.

Suggestion

How to opt in or out of these changes In order for the application to benefit from these changes, it must run on the .NET Framework 4.7.1 or later. The application can benefit from these changes in either of the following ways:

- It is recompiled to target the .NET Framework 4.7.1. These accessibility changes are enabled by default on Windows Forms applications that target the .NET Framework 4.7.1 or later.
- It opts out of the legacy accessibility behaviors by adding the following [AppContext switch](#) to the `<runtime>` section of the app.config file and setting it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
  </startup>
  <runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
    'key1=true/false;key2=true/false' -->
    <AppContextSwitchOverrides
      value="Switch.UseLegacyAccessibilityFeatures=false" />
  </runtime>
</configuration>
```

Applications that target the .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this AppContext switch to `true`.

For an overview of UI automation, see the [UI Automation Overview](#).

Added support for UI Automation patterns and properties

Accessibility clients can take advantage of new WinForms accessibility functionality by using common, publicly described invocation patterns. These patterns are not WinForms-specific. For instance, accessibility clients can call the `QueryInterface` method on the `IAccessible` interface (MAAS) to obtain an `IServiceProvider` interface. If this interface is available, clients can use its `QueryService` method to request an `IAccessibleEx` interface. For more information, see [Using IAccessibleEx from a Client](#). Starting with the .NET Framework 4.7.1, `IServiceProvider` and `IAccessibleEx` (where applicable) are available for WinForms accessibility objects.

The .NET Framework 4.7.1 adds support for the following UI automation patterns and properties:

- The [ToolStripSplitButton](#) and [ComboBox](#) controls support the [Expand/Collapse pattern](#).
- The [ToolStripMenuItem](#) control has a [ControlType](#) property value [ControlType.MenuItem](#).
- The [ToolStripItem](#) control supports the [NameProperty](#) property and the [Expand/Collapse pattern](#).
- The [ToolStripDropDownItem](#) control supports [AccessibleEvents](#) indicating [StateChange](#) and [NameChange](#) when drop down is expanded or collapsed.
- The [ToolStripDropDownButton](#) control has a [ControlType](#) property value of [ControlType.MenuItem](#).
- The [DataGridViewCheckBoxCell](#) control supports the [TogglePattern](#).
- The [NumericUpDown](#) and [DomainUpDown](#) controls support the [NameProperty](#) property and have a [ControlType](#) of [ControlType.Spinner](#).

Improvements to the PropertyGrid control The .NET Framework 4.7.1 adds the following improvements to the PropertyBrowser control:

- The **Details** button in the error dialog that is displayed when the user enters an incorrect value in the [PropertyGrid](#) control supports the [Expand/Collapse pattern](#), state and name change notifications, and a [ControlType](#) property with a value of [ControlType.MenuItem](#).
- The message pane displayed when the **Details** button of the error dialog is expanded is now keyboard accessible and allows Narrator to announce the content of the error message.
- The [AccessibleRole](#) of rows in the [PropertyGrid](#) control have changed from "Row" to "Cell". The cell maps to UIA ControlType "DataItem", which allows it to support appropriate keyboard shortcuts and Narrator announcements.
- The [PropertyGrid](#) control rows that represent header items when the [PropertyGrid](#) control has a [PropertySort](#) property set to [PropertySort.Categorized](#) have a [ControlType](#) property value of [ControlType.Button](#).
- The [PropertyGrid](#) control rows that represent header items when the [PropertyGrid](#) control has a [PropertySort](#) property set to [PropertySort.Categorized](#) support the [Expand/Collapse pattern](#).
- Improved keyboard navigation between the grid and the ToolBar above it. Pressing "Shift-Tab" now selects the first ToolBar button, instead of the whole ToolBar.

- **PropertyGrid** controls displayed in High Contrast mode will now draw a focus rectangle around the ToolBar button which corresponds to the current **PropertySort** property value.
- **PropertyGrid** controls displayed in High Contrast mode and with a **PropertySort** property set to **PropertySort.Categorized** will now display the background of category headers in a highly contrasting color.
- **PropertyGrid** controls better differentiates between ToolBar items with focus and the ToolBar items which indicate the current value of the **PropertySort** property. This fix consists of a High Contrast change and a change for non-High Contrast scenarios.
- **PropertyGrid** control ToolBar items which indicates the current value of the **PropertySort** property support the **TogglePattern**.
- Improved Narrator support for distinguishing the selected alignment in the Alignment Picker.
- When an empty **PropertyGrid** control is displayed on a form, it will now receive focus where previously it would not.

Use of OS-defined colors in High Contrast themes

- The **Button** and **CheckBox** controls with their **FlatStyle** property set to **FlatStyle.System**, which is the default style, now use OS-defined colors in High Contrast theme when selected. Previously, text and background colors were not contrasting and were hard to read.
- The **Button**, **CheckBox**, **RadioButton**, **Label**, **LinkLabel**, and **GroupBox** controls with their **Enabled** property set to **false** used a shaded color to render text in High Contrast themes, resulting in low contrast against the background. Now these controls use the "Disabled Text" color defined by the OS. This fix applies to controls with the **FlatStyle** property set to a value other than **FlatStyle.System**. The latter controls are rendered by the OS.
- **DataGridView** now renders a visible rectangle around the content of the cell which has the current focus. Previously, this was not visible in certain High Contrast themes.
- **ToolStripMenuItem** controls with their **Enabled** property set to **false** now use the "Disabled Text" color defined by the OS.
- **ToolStripMenuItem** controls with their **Checked** property set to **true** now render the associated check mark in a contrasting system color. Previously the check mark color was not contrasting enough and not visible in High Contrast themes. NOTE: Windows 10 has changed values for some high contrast system colors. Windows

Forms Framework is based on the Win32 framework. For the best experience, run on the latest version of Windows and opt in to the latest OS changes by adding an app.manifest file in a test application and un-commenting the following code:

XML

```
<!-- Windows 10 -->
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}" />
```

Improved keyboard navigation

- When a [ComboBox](#) control has its [DropDownStyle](#) property set to [ComboBoxStyle.DropDownList](#) and is the first control in the tab order on the form, it now displays a focus rectangle when the parent form is opened using the keyboard. Before this change, keyboard focus was on this control, but a focus indicator was not rendered.

Improved Narrator support

- The [MonthCalendar](#) control has added support for assistive technologies to access the control, including the ability for Narrator to read the value of the control when previously it could not.
- The [CheckedListBox](#) control now notifies Narrator when a [CheckBox.CheckState](#) property has been changed. Previously, Narrator did not receive notification and as a result users would not be informed that the [CheckState](#) property had been updated.
- The [LinkLabel](#) control has changed the way it notifies Narrator of the text of in the control. Previously, Narrator announced this text twice and read "&" symbols as real text even though they are not visible to a user. The duplicated text was removed from the Narrator announcements, as well as unnecessary "&" symbols.
- The [DataGridViewCell](#) control types now correctly report the read-only status to Narrator and other assistive technologies.
- Narrator is now able to read the System Menu of child windows in [Multiple-Document Interface]~/docs/framework/winforms/advanced/multiple-document-interface-mdi-applications.md) applications.
- Narrator is now able to read [ToolStripMenuItem](#) controls with a [ToolStripItem.Enabled](#) property set to [false](#). Previously, Narrator was unable to focus on disabled menu items to read the content.

Name	Value
Scope	Major
Version	4.8
Type	Retargeting

Affected APIs

- [ToolStripDropDownButton.CreateAccessibilityInstance\(\)](#)
- [DomainUpDown.DomainUpDownAccessibleObject.Name](#)
- [MonthCalendar.AccessibilityObject](#)

Windows Presentation Foundation (WPF)

Accessibility improvements in WPF

Details

High Contrast improvements

- The focus for the [Expander](#) control is now visible. In previous versions of .NET Framework, it was not.
- The text in [CheckBox](#) and [RadioButton](#) controls when they are selected is now easier to see than in previous .NET Framework versions.
- The border of a disabled [ComboBox](#) is now the same color as the disabled text. In previous versions of .NET Framework, it was not.
- Disabled and focused buttons now use the correct theme color. In previous versions of .NET Framework, they did not.
- The dropdown button is now visible when a [ComboBox](#) control's style is set to [ToolBar.ComboBoxStyleKey](#). In previous versions of .NET Framework, it was not.
- The sort indicator arrow in a [DataGrid](#) control now uses theme colors. In previous versions of .NET Framework, it did not.
- The default hyperlink style now changes to the correct theme color on mouse over. In previous versions of .NET Framework, it did not.
- The Keyboard focus on radio buttons is now visible. In previous versions of .NET Framework, it was not.
- The [DataGrid](#) control's checkbox column now uses the expected colors for keyboard focus feedback. In previous versions of .NET Framework, it did not.

- the Keyboard focus visuals are now visible on [ComboBox](#) and [ListBox](#) controls. In previous versions of .NET Framework, it was not.

Screen reader interaction improvements

- [Expander](#) controls are now correctly announced as groups (expand/collapse) by screen readers.
- [DataGridCell](#) controls are now correctly announced as data grid cell (localized) by screen readers.
- Screen readers will now announce the name of an editable [ComboBox](#).
- [PasswordBox](#) controls are no longer announced as "no item in view" by screen readers.

LiveRegion support

Screen readers, such as Narrator, help people understand the user interface (UI) of an application, usually by describing the UI element that currently has focus. However, if a UI element changes somewhere in the screen and it does not have the focus, the user may not be informed and miss important information. LiveRegions are meant to solve this problem. A developer can use them to inform the screen reader or any other [UI Automation](#) client that an important change has been made to a UI element. The screen reader can then decide how and when to inform the user of this change. The [LiveSetting](#) property also lets the screen reader know how important it is to inform the user of the change made to the UI.

Suggestion

How to opt in or out of these changes

In order for the application to benefit from these changes, it must run on .NET Framework 4.7.1 or later. The application can benefit from these changes in either of the following ways:

- Target .NET Framework 4.7.1. This is the recommended approach. These accessibility changes are enabled by default on WPF applications that target .NET Framework 4.7.1 or later.
- It opts out of the legacy accessibility behaviors by adding the following [AppContext Switch](#) in the `<runtime>` section of the app config file and setting it to `false`, as the following example shows.

XML

```

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
  </startup>
  <runtime>
    <!-- ApplicationContextSwitchOverrides value attribute is in the form of
    'key1=true/false;key2=true/false' -->
    <ApplicationContextSwitchOverrides
      value="Switch.UseLegacyAccessibilityFeatures=false" />
  </runtime>
</configuration>

```

Applications that target .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this `ApplicationContext` switch to `true`. For an overview of UI automation, see [UI Automation Overview](#).

Name	Value
Scope	Major
Version	4.7.1
Type	Retargeting

Affected APIs

- `AutomationElementIdentifiers.LiveSettingProperty`
- `AutomationElementIdentifiers.LiveRegionChangedEvent`
- `System.Windows.Automation.AutomationLiveSetting`
- `AutomationProperties.LiveSettingProperty`
- `AutomationProperties.SetLiveSetting(DependencyObject, AutomationLiveSetting)`
- `AutomationProperties.GetLiveSetting(DependencyObject)`
- `AutomationPeer.GetLiveSettingCore()`

Selector SelectionChanged event and SelectedValue property

Details

Starting with the .NET Framework 4.7.1, a `Selector` always updates the value of its `SelectedValue` property before raising the `SelectionChanged` event, when its selection

changes. This makes the `SelectedValue` property consistent with the other selection properties (`SelectedItem` and `SelectedIndex`), which are updated before raising the event.

In the .NET Framework 4.7 and earlier versions, the update to `SelectedValue` happened before the event in most cases, but it happened after the event if the selection change was caused by changing the `SelectedValue` property.

Suggestion

Apps that target the .NET Framework 4.7.1 or later can opt out of this change and use legacy behavior by adding the following to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.TabControl.SelectionPropertiesCanLagBe
hindSelectionChangedEvent=true" />
</runtime>
```

Apps that target the .NET Framework 4.7 or earlier but are running on the .NET Framework 4.7.1 or later can enable the new behavior by adding the following line to the `<runtime>` section of the application .configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.TabControl.SelectionPropertiesCanLagBe
hindSelectionChangedEvent=false" />
</runtime>
```

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Affected APIs

- [TabControl.SelectedContent](#)
- [Selector.SelectionChanged](#)

TabControl SelectionChanged event and SelectedContent property

Details

Starting with the .NET Framework 4.7.1, a [TabControl](#) updates the value of its [SelectedContent](#) property before raising the [SelectionChanged](#) event, when its selection changes. In the .NET Framework 4.7 and earlier versions, the update to SelectedContent happened after the event.

Suggestion

Apps that target the .NET Framework 4.7.1 or later can opt out of this change and use legacy behavior by adding the following to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.TabControl.SelectionPropertiesCanLagBehindSelectionChangedEvent=true" />
</runtime>
```

Apps that target the .NET Framework 4.7 or earlier but are running on the .NET Framework 4.7.1 or later can enable the new behavior by adding the following line to the `<runtime>` section of the application .configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.TabControl.SelectionPropertiesCanLagBehindSelectionChangedEvent=false" />
</runtime>
```

Name	Value
Scope	Minor

Name	Value
Version	4.7.1
Type	Retargeting

Affected APIs

- [TabControl.SelectedContent](#)
- [Selector.SelectionChanged](#)

The default hash algorithm for WPF PackageDigitalSignatureManager is now SHA256

Details

The `System.IO.Packaging.PackageDigitalSignatureManager` provides functionality for digital signatures in relation to WPF packages. In the .NET Framework 4.7 and earlier versions, the default algorithm ([PackageDigitalSignatureManager.DefaultHashAlgorithm](#)) used for signing parts of a package was SHA1. Due to recent security concerns with SHA1, this default has been changed to SHA256 starting with the .NET Framework 4.7.1. This change affects all package signing, including XPS documents.

Suggestion

A developer who wants to utilize this change while targeting a framework version below .NET Framework 4.7.1 or a developer who requires the previous functionality while targeting .NET Framework 4.7.1 or greater can set the following `AppContext` flag appropriately. A value of true will result in SHA1 being used as the default algorithm; false results in SHA256.

XML

```

<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.MS.Internal.UseSha1AsDefaultHashAlgorithmForDigitalSignatures=
        true"/>
  </runtime>
</configuration>

```

Name	Value
Scope	Edge
Version	4.7.1
Type	Retargeting

Affected APIs

- [PackageDigitalSignatureManager.DefaultHashAlgorithm](#)

Windows Workflow Foundation (WF)

Accessibility improvements in Windows Workflow Foundation (WF) workflow designer

Details

The Windows Workflow Foundation (WF) workflow designer is improving how it works with accessibility technologies. These improvements include the following changes:

- The tab order is changed to left to right and top to bottom in some controls:
- The initialize correlation window for setting correlation data for the [InitializeCorrelation](#) activity
- The content definition window for the [Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#) activities
- More functions are available via the keyboard:
- When editing the properties of an activity, property groups can be collapsed by keyboard the first time they are focused.
- Warning icons are now accessible by keyboard.
- The More Properties button in the Properties window is now accessible by keyboard.
- Keyboard users now can access the header items in the Arguments and Variables panes of the Workflow Designer.
- Improved visibility of items with focus, such as when:
- Adding rows to data grids used by the Workflow Designer and activity designers.
- Tabbing through fields in the [ReceiveReply](#) and [SendReply](#) activities.
- Setting default values for variables or arguments
- Screen readers can now correctly recognize:
- Breakpoints set in the workflow designer.

- The [FlowSwitch<T>](#), [FlowDecision](#), and [CorrelationScope](#) activities.
- The contents of the [Receive](#) activity.
- The Target Type for the [InvokeMethod](#) activity.
- The Exception combobox and the Finally section in the [TryCatch](#) activity.
- The Message Type combobox, the splitter in the Add Correlation Initializers window, the Content Definition window, and the CorrelatesOn Defintion window in the messaging activities ([Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#)).
- State machine transitions and transitions destinations.
- Annotations and connectors on [FlowDecision](#) activities.
- The context (right-click) menus for activities.
- The property value editors, the Clear Search button, the By Category and Alphabetical sort buttons, and the Expression Editor dialog in the properties grid.
- The zoom percentage in the Workflow Designer.
- The separator in [Parallel](#) and [Pick](#) activities.
- The [InvokeDelegate](#) activity.
- The Select Types window for dictionary activities
`(Microsoft.Activities.AddToDictionary< TKey , TValue >,
 Microsoft.Activities.RemoveFromDictionary< TKey , TValue >, etc.).`
- The Browse and Select .NET Type window.
- Breadcrumbs in the Workflow Designer.
- Users who choose High Contrast themes will see many improvements in the visibility of the Workflow Designer and its controls like better contrast ratios between elements and more noticeable selection boxes used for focus elements.

Suggestion

If you have an application with a re-hosted workflow designer, your application can benefit from these changes by performing either of these actions:

- Recompile your application to target the .NET Framework 4.7.1. These accessibility changes are enabled by default.
- If your application targets the .NET Framework 4.7 or earlier but is running on the .NET Framework 4.7.1, you can opt out of these legacy accessibility behaviors by adding the following [AppContext switch](#) to the `<runtime>` section of the app.config file and set it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
```

```

</startup>
<runtime>
    <!-- ApplicationContextSwitchOverrides value attribute is in the form of
'key1=true/false;key2=true/false -->
    <ApplicationContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false" />
</runtime>
</configuration>

```

Applications that target the .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this `HttpContext` switch to `true`.

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

.NET Framework 4.7.2

Core

Allow Unicode Bidirectional Control Characters in URIs

Details

Unicode specifies several special control characters used to specify the orientation of text. In previous versions of the .NET Framework, these characters were incorrectly stripped from all URIs even if they were present in their percent-encoded form. In order to better follow [RFC 3987](#), we now allow these characters in URIs. When found unencoded in a URI, they are percent-encoded. When found percent-encoded they are left as-is.

Suggestion

For applications that target versions of .NET Framework starting with 4.7.2, support for Unicode bidirectional characters is enabled by default. If this change is undesirable, you can disable it by adding the following `HttpContextSwitchOverrides` switch to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Uri.DontKeepUnicodeBidiFormattingCharacters=true" />
</runtime>
```

For applications that target earlier versions of the .NET Framework but are running under versions starting with .NET Framework 4.7.2, support for Unicode bidirectional characters is disabled by default. You can enable it by adding the following [AppContextSwitchOverrides](#) switch to the `<runtime>` section of the application configuration file::

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Uri.DontKeepUnicodeBidiFormattingCharacters=false" />
</runtime>
```

Name	Value
Scope	Minor
Version	4.7.2
Type	Retargeting

Affected APIs

- [System.Uri](#)

DeflateStream uses native APIs for decompression

Details

Starting with the .NET Framework 4.7.2, the implementation of decompression in the `T:System.IO.Compression.DeflateStream` class has changed to use native Windows APIs by default. Typically, this results in a substantial performance improvement. All .NET applications targeting the .NET Framework version 4.7.2 or higher use the native implementation. This change might result in some differences in behavior, which include:

- Exception messages may be different. However, the type of exception thrown remains the same.
- Some special situations, such as not having enough memory to complete an operation, may be handled differently.
- There are known differences for parsing gzip header (note: only `GZipStream` set for decompression is affected):
- Exceptions when parsing invalid headers may be thrown at different times.
- The native implementation enforces that values for some reserved flags inside the gzip header (i.e. `FLG` ↗) are set according to the specification, which may cause it to throw an exception where previously invalid values were ignored.

Suggestion

If decompression with native APIs has adversely affected the behavior of your app, you can opt out of this feature by adding the

`Switch.System.IO.Compression.DoNotUseNativeZipLibraryForDecompression` switch to the `runtime` section of your app.config file and setting it to `true`:

XML

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.IO.Compression.DoNotUseNativeZipLibraryForDecompression
      =true" />
  </runtime>
</configuration>
```

Name	Value
Scope	Minor
Version	4.7.2
Type	Retargeting

Affected APIs

- `System.IO.Compression.DeflateStream`
- `System.IO.Compression.GZipStream`

Ensure `System.Uri` uses a consistent reserved character set

Details

In [System.Uri](#), certain percent-encoded characters that were sometimes decoded are now consistently left encoded. This occurs across the properties and methods that access the path, query, fragment, or userinfo components of the URI. The behavior will change only when both of the following are true:

- The URI contains the encoded form of any of the following reserved characters: `:`, `'`, `(`, `)`, `!` or `*`.
- The URI contains a Unicode or encoded non-reserved character. If both of the above are true, the encoded reserved characters are left encoded. In previous versions of the .NET Framework, they are decoded.

Suggestion

For applications that target versions of .NET Framework starting with 4.7.2, the new decoding behavior is enabled by default. If this change is undesirable, you can disable it by adding the following [AppContextSwitchOverrides](#) switch to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Uri.DontEnableStrictRFC3986ReservedCharacterSets=true"
  />
</runtime>
```

For applications that target earlier versions of the .NET Framework but are running under versions starting with .NET Framework 4.7.2, the new decoding behavior is disabled by default. You can enable it by adding the following [AppContextSwitchOverrides](#) switch to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Uri.DontEnableStrictRFC3986ReservedCharacterSets=false"
  />
</runtime>
```

Name	Value
Scope	Minor
Version	4.7.2
Type	Retargeting

Affected APIs

- [System.Uri](#)

Resgen refuses to load content from the web

Details

.resx files may contain binary formatted input. If you attempt to use resgen to load a file that was downloaded from an untrusted location, it will fail to load the input by default.

Suggestion

Resgen users who require loading binary formatted input from untrusted locations can either remove the mark of the web from the input file or apply the opt-out quirk. Add the following registry setting to apply the machine wide opt-out quirk:

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\SDK]

"AllowProcessOfUntrustedResourceFiles"="true"

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Stack traces obtained when using portable PDBs now include source file and line information if requested

Details

Starting with .NET Framework 4.7.2, stack traces obtained when using portable PDBs include source file and line information when requested. In versions prior to .NET

Framework 4.7.2, source file and line information would be unavailable when using portable PDBs even if explicitly requested.

Suggestion

For applications that target the .NET Framework 4.7.2, you can opt out of the source file and line information when using portable PDBs if it is not desirable by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Diagnostics.IgnorePortablePDBsInStackTraces=true" />
</runtime>
```

For applications that target earlier versions of the .NET Framework but run on the .NET Framework 4.7.2 or later, you can opt in to the source file and line information when using portable PDBs by adding the following to the `<runtime>` section of your `app.config` file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Diagnostics.IgnorePortablePDBsInStackTraces=false" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Affected APIs

- [StackTrace\(Boolean\)](#)
- [StackTrace\(Exception, Boolean\)](#)
- [StackTrace\(Exception, Int32, Boolean\)](#)

Windows Forms

Accessibility improvements in Windows Forms controls for .NET 4.7.2

Details

Windows Forms Framework is improving how it works with accessibility technologies to better support Windows Forms customers. These include the following changes:

- Changes to improve display during High Contrast mode.
- Changes to improve the keyboard navigation in the DataGridView and ToolStrip controls.
- Changes to interaction with Narrator.

Suggestion

How to opt in or out of these changes In order for the application to benefit from these changes, it must run on the .NET Framework 4.7.2 or later. The application can benefit from these changes in either of the following ways:

- It is recompiled to target the .NET Framework 4.7.2. These accessibility changes are enabled by default on Windows Forms applications that target the .NET Framework 4.7.2 or later.
- It targets the .NET Framework 4.7.1 or earlier version and opts out of the legacy accessibility behaviors by adding the following [AppContext Switch](#) to the `<runtime>` section of the app config file and setting it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
    </startup>
    <runtime>
        <!-- AppContextSwitchOverrides value attribute is in the form of
        'key1=true/false;key2=true/false' -->
        <AppContextSwitchOverrides
            value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi
            lityFeatures.2=false" />
    </runtime>
</configuration>
```

Note that to opt in to the accessibility features added in .NET Framework 4.7.2, you must also opt in to accessibility features of .NET Framework 4.7.1 as well. Applications that target the .NET Framework 4.7.2 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this `AppContext` switch to `true`.

Use of OS-defined colors in High Contrast themes

- The drop down arrow of the `ToolStripDropDownButton` now uses OS-defined colors in High Contrast theme.
- `Button`, `RadioButton` and `CheckBox` controls with `FlatStyle` set to `FlatStyle.Flat` or `FlatStyle.Popup` now use OS-defined colors in High Contrast theme when selected. Previously, text and background colors were not contrasting and were hard to read.
- Controls contained within a `GroupBox` that has its `Enabled` property set to `false` will now use OS-defined colors in High Contrast theme.
- The `ToolStripButton`, `ToolStripComboBox`, and `ToolStripDropDownButton` controls have an increased luminosity contrast ratio in High Contrast Mode.
- `DataGridViewLinkCell` will by default use OS-defined colors in High Contrast mode for the `DataGridViewLinkCell.LinkColor` property. NOTE: Windows 10 has changed values for some high contrast system colors. Windows Forms Framework is based on the Win32 framework. For the best experience, run on the latest version of Windows and opt in to the latest OS changes by adding an app.manifest file in a test application and un-commenting the following code:

XML

```
<!-- Windows 10 -->
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}" />
```

Improved Narrator support

- Narrator now announces the value of the `ToolStripMenuItem.ShortcutKeys` property when announcing the text of a `ToolStripMenuItem`.
- Narrator now indicates when a `ToolStripMenuItem` has its `Enabled` property set to `false`.
- Narrator now gives feedback on the state of a check box when the `ListView.CheckBoxes` property is set to `true`.
- Narrator Scan Mode focus order is now consistent with the visual order of the controls on the ClickOnce download dialog window.

Improved DataGridView Accessibility support

- Rows in a [DataGridView](#) can now be sorted using the keyboard. Now a user can use the F3 key in order to sort by the current column.
- When the [DataGridView.SelectionMode](#) is set to [DataGridViewSelectionMode.FullRowSelect](#), the column header will change color to indicate the current column as the user tabs through the cells in the current row.
- The [DataGridViewCell.DataGridViewCellAccessibleObject.Parent](#) property now returns the correct parent control.

Improved Visual cues

- The [RadioButton](#) and [CheckBox](#) controls with an empty [Text](#) property will now display a focus indicator when they receive focus.

Improved Property Grid Support

- The [PropertyGrid](#) control child elements now return a `true` for the [IsReadOnlyProperty](#) property only when a [PropertyGrid](#) element is enabled.
- The [PropertyGrid](#) control child elements now return a `false` for the [IsEnabledProperty](#) property only when a [PropertyGrid](#) element can be changed by the user. For an overview of UI automation, see the [UI Automation Overview](#).

Improved keyboard navigation

- [ToolStripButton](#) now allows focus when contained within a [ToolStripPanel](#) that has the [TabStop](#) property set to `true`.

Name	Value
Scope	Major
Version	4.7.2
Type	Retargeting

ContextMenuStrip.SourceControl property contains a valid control in the case of nested ToolStripMenuItem

Details

In the .NET Framework 4.7.1 and previous versions, the [ContextMenuStrip.SourceControl](#) property incorrectly returns null when the user opens the menu from nested [ToolStripMenuItem](#) controls. In the .NET Framework 4.7.2 and later, [SourceControl](#) property is always set to the actual source control.

Suggestion

How to opt in or out of these changes In order for an application to benefit from these changes, it must run on the .NET Framework 4.7.2 or later. The application can benefit from these changes in either of the following ways:

- It targets the .NET Framework 4.7.2. This change is enabled by default on Windows Forms applications that target the .NET Framework 4.7.2 or later.
- It targets the .NET Framework 4.7.1 or an earlier version and opts out of the legacy accessibility behaviors by adding the following [AppContext Switch](#) to the `<runtime>` section of the app.config file and setting it to `false`, as the following example shows.

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Windows.Forms.UseLegacyContextMenuSourceControlValue=false"/>
</runtime>
```

Applications that target the .NET Framework 4.7.2 or later, and want to preserve the legacy behavior can opt in to the use of the legacy source control value by explicitly setting this AppContext switch to `true`.

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Affected APIs

- [ContextMenuStrip.SourceControl](#)

PrivateFontCollection.AddFontFile method releases Font resources

Details

In the .NET Framework 4.7.1 and previous versions, the [System.Drawing.Text.PrivateFontCollection](#) class does not release the GDI+ font resources after the [PrivateFontCollection](#) is disposed for [Font](#) objects that are added to this collection using the [AddFontFile\(String\)](#) method. In the .NET Framework 4.7.2 and later [Dispose](#) releases the GDI+ fonts that were added to the collection as files.

Suggestion

How to opt in or out of these changes In order for an application to benefit from these changes, it must run on the .NET Framework 4.7.2 or later. The application can benefit from these changes in either of the following ways:

- It is recompiled to target the .NET Framework 4.7.2. This change is enabled by default on Windows Forms applications that target the .NET Framework 4.7.2 or later.
- It targets the .NET Framework 4.7.1 or an earlier version and opts out of the legacy accessibility behaviors by adding the following [AppContext Switch](#) to the `<runtime>` section of the app.config file and setting it to `false`, as the following example shows.

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Drawing.Text.DoNotRemoveGdiFontsResourcesFromFontCollection=false"/>
</runtime>
```

Applications that target the .NET Framework 4.7.2 or later, and want to preserve the legacy behavior can opt in to not release font resources by explicitly setting this AppContext switch to `true`.

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Affected APIs

- [PrivateFontCollection.AddFontFile\(String\)](#)

- `FontCollection.Dispose()`

WinForm's Domain upbutton and downbutton actions are in sync now

Details

In the .NET Framework 4.7.1 and previous versions the [DomainUpDown](#) control's [DomainUpDown.UpButton\(\)](#) action is ignored when control text is present, and the developer is required to use [DomainUpDown.DownButton\(\)](#) action on the control before using [DomainUpDown.UpButton\(\)](#) action. Starting with the .NET Framework 4.7.2 both the [DomainUpDown.UpButton\(\)](#) and [DomainUpDown.DownButton\(\)](#) actions work independently in this scenario and remain in sync.

Suggestion

In order for an application to benefit from these changes, it must run on the .NET Framework 4.7.2 or later. The application can benefit from these changes in either of the following ways:

- It is recompiled to target the .NET Framework 4.7.2. This change is enabled by default on Windows Forms applications that target the .NET Framework 4.7.2 or later.
- It opts out of the legacy scrolling behavior by adding the following [AppContext Switch](#) to the `<runtime>` section of the app config file and setting it to `false`, as the following example shows.

XML

```
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Forms.DomainUpDown.UseLegacyScrolling=false"/>
</runtime>
```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Affected APIs

- `DomainUpDown.UpButton()`
- `DomainUpDown.DownButton()`

Windows Presentation Foundation (WPF)

Keyboard focus now moves correctly across multiple layers of WinForms/WPF hosting

Details

Consider a WPF application hosting a WinForms control which in turn hosts WPF controls. Users may not be able to tab out of the WinForms layer if the first or last control in that layer is the WPF `System.Windows.Forms.Integration.ElementHost`. This change fixes this issue, and users are now able to tab out of the WinForms layer. Automated applications that rely on focus never escaping the WinForms layer may no longer work as expected.

Suggestion

A developer who wants to utilize this change while targeting a framework version below .NET 4.7.2 can set the following set of `AppContext` flags to false for the change to be enabled.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi
lityFeatures.2=false"/>
</runtime>
</configuration>
```

WPF applications must opt in to all early accessibility improvements to get the later improvements. In other words, both the `Switch.UseLegacyAccessibilityFeatures` and the `Switch.UseLegacyAccessibilityFeatures.2` switches must be set. A developer who requires the previous functionality while targeting .NET 4.7.2 or greater can set the following `AppContext` flag to true for the change to be disabled.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures.2=true"/>
</runtime>
</configuration>
```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

The default hash algorithm for WPF's Markup Compiler is now SHA256

Details

The WPF MarkupCompiler provides compilation services for XAML markup files. In the .NET Framework 4.7.1 and earlier versions, the default hash algorithm used for checksums was SHA1. Due to recent security concerns with SHA1, this default has been changed to SHA256 starting with the .NET Framework 4.7.2. This change affects all checksum generation for markup files during compilation.

Suggestion

A developer who targets .NET Framework 4.7.2 or greater and wants to revert to SHA1 hashing behavior must set the following AppContext flag.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Markup.DoNotUseSha256ForMarkupCompilerChecksumA
lgorithm=true"/>
</runtime>
</configuration>
```

A developer who wants to utilize SHA256 hashing while targeting a framework version below .NET 4.7.2 must set the below AppContext flag. Note that the installed version of

the .NET Framework must be 4.7.2 or greater.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Markup.DoNotUseSha256ForMarkupCompilerChecksumAlgorithm=false"
/>
</runtime>
</configuration>
```

Name	Value
Scope	Transparent
Version	4.7.2
Type	Retargeting

WPF AppDomain Shutdown Handling May Now Call Dispatcher.Invoke in Cleanup of Weak Events

Details

In .NET Framework 4.7.1 and earlier versions, WPF potentially creates a [System.Windows.Threading.Dispatcher](#) on the .NET finalizer thread during AppDomain shutdown. This was fixed in .NET Framework 4.7.2 and later versions by making the cleanup of weak events thread-aware. Due to this, WPF may call [Dispatcher.Invoke](#) to complete the cleanup process. In certain applications, this change in finalizer timing can potentially cause exceptions during AppDomain or process shutdown. This is generally seen in applications that do not correctly shut down dispatchers running on worker threads prior to process or AppDomain shutdown. Such applications should take care to properly manage the lifetime of dispatchers.

Suggestion

In .NET Framework 4.7.2 and later versions, developers can disable this fix in order to help alleviate (but not eliminate) timing issues that may occur due to the cleanup change. To disable the change in cleanup, use the following AppContext flag.

XML

```

<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.MS.Internal.DoNotInvokeInWeakEventTableShutdownListener=true" />
</runtime>
</configuration>

```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

WPF Changing a primary key when displaying ADO data in a Master/Detail scenario

Details

Suppose you have an ADO collection of items of type `Order`, with a relation named "OrderDetails" relating it to a collection of items of type `Detail` via the primary key "OrderID". In your WPF app, you can bind a list control to the details for a given order:

XAML

```
<ListBox ItemsSource="{Binding Path=OrderDetails}" >
```

where the DataContext is an `Order`. WPF gets the value of the `OrderDetails` property - a collection D of all the `Detail` items whose `OrderID` matches the `OrderID` of the master item. The behavior change arises when you change the primary key `OrderID` of the master item. ADO automatically changes the `OrderID` of each of the affected records in the Details collection (namely the ones copied into collection D). But what happens to D?

- Old behavior: Collection D is cleared. The master item does *not* raise a change notification for property `OrderDetails`. The ListBox continues to use collection D, which is now empty.
- New behavior: Collection D is unchanged. Each of its items raises a change notification for the `OrderID` property. The ListBox continues to use collection D, and displays the details with the new `OrderID`. WPF implements the new behavior

by creating collection D in a different way: by calling the ADO method `DataRowView.CreateChildView(DataRelation, Boolean)` with the `followParent` argument set to `true`.

Suggestion

An app gets the new behavior by using the following AppContext switch.

```
XML

<configuration>
  <runtime>
    <AppContextSwitchOverrides
      value="Switch.System.Windows.Data.DoNotUseFollowParentWhenBindingToADODataRe
      lation=false"/>
  </runtime>
</configuration>
```

The switch defaults to `true` (old behavior) for apps that target .NET 4.7.1 or below, and to `false` (new behavior) for apps that target .NET 4.7.2 or above.

Name	Value
Scope	Minor
Version	4.7.2
Type	Retargeting

WPF FocusVisual for RadioButton and CheckBox Now Displays Correctly When The Controls Have No Content

Details

In the .NET Framework 4.7.1 and earlier versions, WPF `System.Windows.Controls.CheckBox` and `System.Windows.Controls.RadioButton` have inconsistent and, in Classic and High Contrast themes, incorrect focus visuals. These issues occur in cases where the controls do not have any content set. This can make the transition between themes confusing and the focus visual hard to see. In the .NET Framework 4.7.2, these visuals are now more consistent across themes and more easily visible in Classic and High Contrast themes.

Suggestion

A developer targeting .NET Framework 4.7.2 that wants to revert to the behavior in .NET 4.7.1 will need to set the following ApplicationContext flag.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures.2=true;" />
</runtime>
</configuration>
```

A developer who wants to utilize this change while targeting a framework version below .NET 4.7.2 must set the following ApplicationContext flags. Note that all the flags must be set appropriately and the installed version of the .NET Framework must be 4.7.2 or greater. WPF applications are required to opt in to all earlier accessibility improvements to get the latest improvements. To do this, ensure that both the ApplicationContext switches 'Switch.UseLegacyAccessibilityFeatures' and 'Switch.UseLegacyAccessibilityFeatures.2' are set to false.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibilityFeatures.2=false;" />
</runtime>
</configuration>
```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

WPF TextBox/PasswordBox Text Selection Does Not Follow System Colors

Details

In .NET Framework 4.7.1 and earlier versions, WPF `System.Windows.Controls.TextBox` and `System.Windows.Controls.PasswordBox` could only render a text selection in the Adorner layer. In some system themes this would occlude text, making it hard to read. In .NET Framework 4.7.2 and later, developers have an option of enabling a non-Adorner-based selection rendering scheme that alleviates this issue.

Suggestion

A developer who wants to utilize this change must set the following AppContext flag appropriately. To utilize this feature, the installed .NET Framework version must be 4.7.2 or greater. To enable the non-adorner-based selection, use the following AppContext flag.

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Windows.Controls.Text.UseAdornerForTextboxSelectionRend
ering=false"/>
</runtime>
</configuration>
```

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Windows Workflow Foundation (WF)

**Avoiding endless recursion for
IWorkflowInstanceManagement.TransactedCancel and
IWorkflowInstanceManagement.TransactedTerminate**

Details

Under some circumstances when using `IWorkflowInstanceManagement.TransactedCancel` or `IWorkflowInstanceManagement.TransactedTerminate` APIs to cancel or terminate a

workflow service instance, the workflow instance may encounter a stack overflow due to endless recursion when the `Workflow` runtime attempts to persist the service instance as part of processing the request. The problem occurs if the workflow instance is in a state where it is waiting for some other outstanding WCF request to another service to complete. The `TransactedCancel` and `TransactedTerminate` operations create work items that are queued for the workflow service instance. These work items are not executed as part of the processing of the `TransactedCancel/TransactedTerminate` request. Because the workflow service instance is busy waiting for the other outstanding WCF request to complete, the work item created remains queued. The `TransactedCancel/TransactedTerminate` operation completes and control is returned back to the client. When the transaction associated with the `TransactedCancel/TransactedTerminate` operation attempts to commit, it needs to persist the workflow service instance state. But because there is an outstanding `WCF` request for the instance, the Workflow runtime cannot persist the workflow service instance, and an endless recursion loop leads to the stack overflow. Because `TransactedCancel` and `TransactedTerminate` only create a work item in memory, the fact that a transaction exists doesn't have any effect. A rollback of the transaction does not discard the work item. To address this issue, starting in .NET Framework 4.7.2, we have introduced an `AppSetting` that can be added to the `web.config/app.config` of the workflow service that tells it to ignore transactions for `TransactedCancel` and `TransactedTerminate`. This allows the transaction to commit without waiting for the workflow instance to persist. The AppSetting for this feature is named `microsoft:WorkflowServices:IgnoreTransactionsForTransactedCancelAndTransactedTerminate`. A value of `true` indicates that the transaction should be ignored, thus avoiding the stack overflow. The default value of this AppSetting is `false`, so existing workflow service instances are not affected.

Suggestion

If you are using AppFabric or another `IWorkflowInstanceManagement` client and are encountering a stack overflow in the workflow service instance when trying to cancel or terminate a workflow instance, you can add the following to the `<appSettings>` section of the `web.config/app.config` file for the workflow service:

XML

```
<add  
key="microsoft:WorkflowServices:IgnoreTransactionsForTransactedCancelAndTran  
sactedTerminate" value="true"/>
```

If you are not encountering the problem, you do not need to do this.

Name	Value
Scope	Edge
Version	4.7.2
Type	Retargeting

Retargeting changes for migration to .NET Framework 4.8.x

Article • 08/10/2023

This article lists the app compatibility issues that were introduced in .NET Framework 4.8 and 4.8.1.

.NET Framework 4.8

Core

Managed cryptography classes do not throw a CryptographyException in FIPS mode

Details

In .NET Framework 4.7.2 and earlier versions, managed cryptographic provider classes such as [SHA256Managed](#) throw a [CryptographicException](#) when the system cryptographic libraries are configured in FIPS mode. These exceptions are thrown because the managed versions have not undergone FIPS (Federal Information Processing Standards) 140-2 certification, as well as to block cryptographic algorithms that were not considered to be approved based on the FIPS rules. Because few developers have their development machines in FIPS mode, these exceptions are frequently thrown only on production systems. Applications that target .NET Framework 4.8 and later versions automatically switch to the newer, relaxed policy, so that a [CryptographicException](#) is no longer thrown by default in such cases. Instead, the managed cryptography classes redirect cryptographic operations to a system cryptography library. This policy change effectively removes a potentially confusing difference between developer environments and the production environments and makes native components and managed components operate under the same cryptographic policy.

Suggestion

If this behavior is undesirable, you can opt out of it and restore the previous behavior so that a [CryptographicException](#) is thrown in FIPS mode by adding the following [AppContextSwitchOverrides](#) configuration setting to the [`<runtime>`](#) section of your application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Security.Cryptography.UseLegacyFipsThrow=true" />
</runtime>
```

If your application targets .NET Framework 4.7.2 or earlier, you can also opt in to this change by adding the following `AppContextSwitchOverrides` configuration setting to the `<runtime>` section of your application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Security.Cryptography.UseLegacyFipsThrow=false" />
</runtime>
```

Name	Value
Scope	Edge
Version	4.8
Type	Retargeting

Affected APIs

- `System.Security.Cryptography.AesManaged`
- `System.Security.Cryptography.MD5Cng`
- `System.Security.Cryptography.MD5CryptoServiceProvider`
- `System.Security.Cryptography.RC2CryptoServiceProvider`
- `System.Security.Cryptography.RijndaelManaged`
- `System.Security.Cryptography.RIPEMD160Managed`
- `System.Security.Cryptography.SHA1Managed`
- `System.Security.Cryptography.SHA256Managed`

Windows Forms

Accessibility improvements in Windows Forms controls for .NET 4.8

Details

The Windows Forms Framework is continuing to improve how it works with accessibility technologies to better support Windows Forms customers. These include the following changes:

- Changes to improve display during High Contrast mode.
- Changes to interaction with Narrator.
- Changes in the Accessible hierarchy (improving navigation through the UI Automation tree).

Suggestion

How to opt in or out of these changes In order for the application to benefit from these changes, it must run on the .NET Framework 4.8. The application can opt in into these changes in either of the following ways:

- It is recompiled to target the .NET Framework 4.8. These accessibility changes are enabled by default on Windows Forms applications that target the .NET Framework 4.8.
- It targets the .NET Framework 4.7.2 or earlier version and opts out of the legacy accessibility behaviors by adding the following [AppContext switch](#) to the `<runtime>` section of the app config file and setting it to `false`, as the following example shows.

```
XML

<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
  </startup>
  <runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
    'key1=true/false;key2=true/false' -->
    <AppContextSwitchOverrides
      value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibilityFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false" />
  </runtime>
</configuration>
```

Note that to opt in to the accessibility features added in .NET Framework 4.8, you must also opt in to accessibility features of .NET Framework 4.7.1 and 4.7.2 as well. Applications that target the .NET Framework 4.8 and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this AppContext switch to `true`. Enabling the keyboard ToolTip invocation support requires adding the `Switch.System.Windows.Forms.UseLegacyToolTipDisplay=false` line to the `AppContextSwitchOverrides` value:

XML

```
<AppContextSwitchOverrides  
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibilit  
yFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false;Switch.System.W  
indows.Forms.UseLegacyToolTipDisplay=false" />
```

Note that enabling this feature requires opting in to the aforementioned accessibility features of .NET Framework 4.7.1 - 4.8. Also, if any of the accessibility features are not opted in but the tooltip display feature is opted in, a runtime [NotSupportedException](#) will be thrown on the first access to these features. The exception message indicates that keyboard ToolTips require accessibility improvements of level 3 to be enabled.

Use of OS-defined colors in High Contrast themes

- Improved high-contrast themes.

Improved Narrator support

- Narrator now announces the sort direction of the [DataGridViewColumn](#) when announcing an accessible name of a [DataGridViewCell](#).

Improved CheckedListBox Accessibility support

- Improved Narrator support for the [CheckedListBox](#) control. When navigating to the [CheckedListBox](#) control using the keyboard, Narrator focuses the [CheckedListBox](#) item and announces it.
- An empty CheckedListBox control now has a focus rectangle drawn for a virtual first item when the control becomes focused.

Improved ComboBox Accessibility support

- Enabled UI Automation support for the [ComboBox](#) control, with the ability to use UI Automation notifications and other UI Automation features. [Improved DataGridView Accessibility support](#)
- Enabled UI Automation support for [DataGridView](#) control with ability to use UI Automation notifications and other UI Automation features.
- The UI Automation element which corresponds to the [DataGridViewComboBoxEditingControl](#) or [DataGridViewTextBoxEditingControl](#) is now a child of corresponding editing cell.

Improved LinkLabel Accessibility support

- Improved [LinkLabel](#) control accessibility: Narrator announces the disabled state for the link if the corresponding [LinkLabel](#) control is disabled.

Improved ProgressBar Accessibility support

- Enabled UI Automation support for the [ProgressBar](#) control with the ability to use UI Automation notifications and other UI Automation features. Developers are now able to use UI Automation notifications which Narrator can announce to indicate progress. For an overview of UI automation events overview, including UI automation notification events, see the [UI Automation Events Overview](#).

Improved PropertyGrid Accessibility support

- Enabled UI Automation support for the [PropertyGrid](#) control, with the ability to use UI Automation notifications and other UI Automation features.
- The UI Automation element which corresponds to the currently edited property is now a child of the corresponding property item UI Automation element.
- The UI Automation property item element is now a child of the corresponding category element if the parent [PropertyGrid](#) control is set to category view.

Improved ToolStrip support

- Enabled UI Automation support for the [ToolStrip](#) control, with the ability to use UI Automation notifications and other UI Automation features.
- Improved navigation through [ToolStrip](#) items.
- In items mode, Narrator focus does not disappear and does not go to hidden items.

Improved Visual cues

- An empty [CheckedListBox](#) control now displays a focus indicator when it receives focus. Note: UI automation support is enabled for controls in runtime but is not used in design time. For an overview of UI automation, see the [UI Automation Overview](#).

Invoking controls' ToolTips with a keyboard

- Control tooltip can now be invoked by focusing the control with keyboard. This feature needs to be enabled explicitly for the application (see section "[How to opt in or out of these changes](#)")

Name	Value
Scope	Major
Version	4.8
Type	Retargeting

Windows Presentation Foundation (WPF)

Accessibility improvements in WPF

Details

High Contrast improvements

- The focus for the [Expander](#) control is now visible. In previous versions of .NET Framework, it was not.
- The text in [CheckBox](#) and [RadioButton](#) controls when they are selected is now easier to see than in previous .NET Framework versions.
- The border of a disabled [ComboBox](#) is now the same color as the disabled text. In previous versions of .NET Framework, it was not.
- Disabled and focused buttons now use the correct theme color. In previous versions of .NET Framework, they did not.
- The dropdown button is now visible when a [ComboBox](#) control's style is set to [ToolBar.ComboBoxStyleKey](#). In previous versions of .NET Framework, it was not.
- The sort indicator arrow in a [DataGrid](#) control now uses theme colors. In previous versions of .NET Framework, it did not.
- The default hyperlink style now changes to the correct theme color on mouse over. In previous versions of .NET Framework, it did not.
- The Keyboard focus on radio buttons is now visible. In previous versions of .NET Framework, it was not.
- The [DataGrid](#) control's checkbox column now uses the expected colors for keyboard focus feedback. In previous versions of .NET Framework, it did not.
- the Keyboard focus visuals are now visible on [ComboBox](#) and [ListBox](#) controls. In previous versions of .NET Framework, it was not.

Screen reader interaction improvements

- [Expander](#) controls are now correctly announced as groups (expand/collapse) by screen readers.
- [DataGridCell](#) controls are now correctly announced as data grid cell (localized) by screen readers.
- Screen readers will now announce the name of an editable [ComboBox](#).
- [PasswordBox](#) controls are no longer announced as "no item in view" by screen readers.

LiveRegion support

Screen readers, such as Narrator, help people understand the user interface (UI) of an application, usually by describing the UI element that currently has focus. However, if a UI element changes somewhere in the screen and it does not have the focus, the user may not be informed and miss important information. LiveRegions are meant to solve this

problem. A developer can use them to inform the screen reader or any other UI Automation client that an important change has been made to a UI element. The screen reader can then decide how and when to inform the user of this change. The `LiveSetting` property also lets the screen reader know how important it is to inform the user of the change made to the UI.

Suggestion

How to opt in or out of these changes

In order for the application to benefit from these changes, it must run on .NET Framework 4.7.1 or later. The application can benefit from these changes in either of the following ways:

- Target .NET Framework 4.7.1. This is the recommended approach. These accessibility changes are enabled by default on WPF applications that target .NET Framework 4.7.1 or later.
- It opts out of the legacy accessibility behaviors by adding the following `AppContextSwitch` in the `<runtime>` section of the app config file and setting it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
  </startup>
  <runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
    'key1=true/false;key2=true/false' -->
    <AppContextSwitchOverrides
      value="Switch.UseLegacyAccessibilityFeatures=false" />
  </runtime>
</configuration>
```

Applications that target .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this `AppContext` switch to `true`. For an overview of UI automation, see [UI Automation Overview](#).

Name	Value
Scope	Major
Version	4.7.1

Name	Value
Type	Retargeting

Affected APIs

- [AutomationElementIdentifiers.LiveSettingProperty](#)
- [AutomationElementIdentifiers.LiveRegionChangedEvent](#)
- [System.Windows.Automation.AutomationLiveSetting](#)
- [AutomationProperties.LiveSettingProperty](#)
- [AutomationProperties.SetLiveSetting\(DependencyObject, AutomationLiveSetting\)](#)
- [AutomationProperties.GetLiveSetting\(DependencyObject\)](#)
- [AutomationPeer.GetLiveSettingCore\(\)](#)

Add SelectionTextBrush public property to TextBox/PasswordBox non-adorner selection

Details

In WPF applications using [non-adorner based text selection](#) for [TextBox](#) and [PasswordBox](#), developers may now set the newly added SelectionTextBrush property in order to alter the rendering of the selected text. By default, this color changes with [HighlightTextBrushKey](#). If non-adorner based text selection is not enabled, this property does nothing.

Suggestion

Once non-adorner based text selection is enabled, you can use the [PasswordBox.SelectionTextBrush](#) and [SelectionTextBrush](#) property to change the appearance of the selected text. This can be achieved using XAML:

XAML
<pre><TextBox SelectionBrush="Red" SelectionTextBrush="White" SelectionOpacity="0.5" Foreground="Blue" CaretBrush="Blue"> This is some text. </TextBox></pre>

Name	Value
Scope	Major

Name	Value
Version	4.8
Type	Retargeting

Affected APIs

- [TextBoxBase.SelectionTextBrushProperty](#)
- [TextBoxBase.SelectionTextBrush](#)
- [System.Windows.Controls.TextBox](#)
- [System.Windows.Controls.PasswordBox](#)

HwndHost now correctly resizes child-HWND during DPI changes

Details

In .NET Framework 4.7.2 and earlier versions, when WPF was run in Per-Monitor Aware mode, controls hosted within [HwndHost](#) were not sized correctly after DPI changes, such as when moving applications from one monitor to another. This fix ensures that hosted controls are sized appropriately.

Suggestion

In order for the application to benefit from these changes, it must run on the .NET Framework 4.7.2 or later, and it must opt-in to this behavior by setting the following [AppContext Switch](#) in the `<runtime>` section of the app config file to `false`, as the following example shows.

```
XML

<?xml version="1.0" encoding="utf-8"?>
<configuration>
<startup>
<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
</startup>
<runtime>
<!-- AppContextSwitchOverrides value attribute is in the form of
'key1=true/false;key2=true/false' -->
<AppContextSwitchOverrides
value="Switch.System.Windows.DoNotUsePresentationDpiCapabilityTier2OrGreater=false" />
```

```
</runtime>  
</configuration>
```

Name	Value
Scope	Major
Version	4.8
Type	Retargeting

Windows Workflow Foundation (WF)

Accessibility improvements in Windows Workflow Foundation (WF) workflow designer

Details

The Windows Workflow Foundation (WF) workflow designer is improving how it works with accessibility technologies. These improvements include the following changes:

- The tab order is changed to left to right and top to bottom in some controls:
- The initialize correlation window for setting correlation data for the [InitializeCorrelation](#) activity
- The content definition window for the [Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#) activities
- More functions are available via the keyboard:
- When editing the properties of an activity, property groups can be collapsed by keyboard the first time they are focused.
- Warning icons are now accessible by keyboard.
- The More Properties button in the Properties window is now accessible by keyboard.
- Keyboard users now can access the header items in the Arguments and Variables panes of the Workflow Designer.
- Improved visibility of items with focus, such as when:
- Adding rows to data grids used by the Workflow Designer and activity designers.
- Tabbing through fields in the [ReceiveReply](#) and [SendReply](#) activities.
- Setting default values for variables or arguments
- Screen readers can now correctly recognize:
- Breakpoints set in the workflow designer.
- The [FlowSwitch<T>](#), [FlowDecision](#), and [CorrelationScope](#) activities.
- The contents of the [Receive](#) activity.
- The Target Type for the [InvokeMethod](#) activity.

- The Exception combobox and the Finally section in the [TryCatch](#) activity.
- The Message Type combobox, the splitter in the Add Correlation Initializers window, the Content Definition window, and the CorrelatesOn Defintion window in the messaging activities ([Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#)).
- State machine transitions and transitions destinations.
- Annotations and connectors on [FlowDecision](#) activities.
- The context (right-click) menus for activities.
- The property value editors, the Clear Search button, the By Category and Alphabetical sort buttons, and the Expression Editor dialog in the properties grid.
- The zoom percentage in the Workflow Designer.
- The separator in [Parallel](#) and [Pick](#) activities.
- The [InvokeDelegate](#) activity.
- The Select Types window for dictionary activities
`(Microsoft.Activities.AddToDictionary< TKey, TValue >, Microsoft.Activities.RemoveFromDictionary< TKey, TValue >, etc.).`
- The Browse and Select .NET Type window.
- Breadcrumbs in the Workflow Designer.
- Users who choose High Contrast themes will see many improvements in the visibility of the Workflow Designer and its controls like better contrast ratios between elements and more noticeable selection boxes used for focus elements.

Suggestion

If you have an application with a re-hosted workflow designer, your application can benefit from these changes by performing either of these actions:

- Recompile your application to target the .NET Framework 4.7.1. These accessibility changes are enabled by default.
- If your application targets the .NET Framework 4.7 or earlier but is running on the .NET Framework 4.7.1, you can opt out of these legacy accessibility behaviors by adding the following [AppContext switch](#) to the `<runtime>` section of the app.config file and set it to `false`, as the following example shows.

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7"/>
  </startup>
  <runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
    'key1=true/false;key2=true/false' -->
    <AppContextSwitchOverrides
      value="Switch.UseLegacyAccessibilityFeatures=false" />
  </runtime>
</configuration>
```

```
</runtime>
</configuration>
```

Applications that target the .NET Framework 4.7.1 or later and want to preserve the legacy accessibility behavior can opt in to the use of legacy accessibility features by explicitly setting this `AppContext` switch to `true`.

Name	Value
Scope	Minor
Version	4.7.1
Type	Retargeting

Workflow XAML checksums for symbols changed from SHA1 to SHA256

Details

To support debugging with Visual Studio, the Workflow runtime generates a checksum for a workflow XAML file using a hashing algorithm. In the .NET Framework 4.6.2 and earlier versions, workflow checksum hashing used the MD5 algorithm, which caused issues on FIPS-enabled systems. Starting with the .NET Framework 4.7, the default algorithm was changed to SHA1. Starting with the .NET Framework 4.8, the default algorithm was changed to SHA256.

Suggestion

If your code is unable to load workflow instances or to find appropriate symbols due to a checksum failure, try setting the `AppContext` switch "Switch.System.Activities.UseSHA1HashForDebuggerSymbols" to `true`. In code:

C#

```
System.AppContext.SetSwitch("Switch.System.Activities.UseSHA1HashForDebuggerSymbols", true);
```

Or in configuration:

XML

```
<configuration>
  <runtime>
```

```

<AppContextSwitchOverrides
  value="Switch.System.Activities.UseSHA1HashForDebuggerSymbols=true" />
</runtime>
</configuration>

```

Name	Value
Scope	Minor
Version	4.8
Type	Retargeting

Workflow XOML definition and SqlTrackingService cache keys changed from MD5 to SHA256

Details

The Workflow Runtime keeps a cache of workflow definitions defined in XOML. The SqlTrackingService also keeps a cache that is keyed by strings. These caches are keyed by values that include checksum hash value. In the .NET Framework 4.7.2 and earlier versions, this checksum hashing used the MD5 algorithm, which caused issues on FIPS-enabled systems. Starting with the .NET Framework 4.8, the algorithm used is SHA256. There shouldn't be a compatibility issue with this change because the values are recalculated each time the Workflow Runtime and SqlTrackingService is started. However, we have provided quirks to allow customers to revert back to usage of the legacy hashing algorithm, if necessary.

Suggestion

If this change presents a problem when executing workflows, try setting one or both of the `AppContext` switches:

- "Switch.System.Workflow.Runtime.UseLegacyHashForWorkflowDefinitionDispenserCacheKey" to true.
- "Switch.System.Workflow.Runtime.UseLegacyHashForSqlTrackingCacheKey" to true. In code:

C#

```

System.AppContext.SetSwitch("Switch.System.Workflow.Runtime.UseLegacyHashForWorkflowDefinitionDispenserCacheKey", true);
System.AppContext.SetSwitch("Switch.System.Workflow.Runtime.UseLegacyHashForSqlTrackingCacheKey", true);

```

Or in the configuration file (this needs to be in the config file for the application that is creating the [WorkflowRuntime](#) object):

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Workflow.Runtime.UseLegacyHashForWorkflowDefinitionDispenserCacheKey=true" />
<AppContextSwitchOverrides
value="Switch.System.Workflow.Runtime.UseLegacyHashForSqlTrackingCacheKeytrue"
/>
</runtime>
</configuration>
```

Name	Value
Scope	Minor
Version	4.8
Type	Retargeting

Workflow XOML file checksums changed from MD5 to SHA256

Details

To support debugging XOML-based workflows with Visual Studio, when workflow projects containing XOML files build, a checksum of the contents of the XOML file is included in the code generated as a [WorkflowMarkupSourceAttribute.MD5Digest](#) value. In the .NET Framework 4.7.2 and earlier versions, this checksum hashing used the MD5 algorithm, which caused issues on FIPS-enabled systems. Starting with the .NET Framework 4.8, the algorithm used is SHA256. To be compatible with the [WorkflowMarkupSourceAttribute.MD5Digest](#), only the first 16 bytes of the generated checksum are used. This may cause problems during debugging. You may need to re-build your project.

Suggestion

If re-building your project does not solve the problem, try setting the [AppContext](#) switch "Switch.System.Workflow.ComponentModel.UseLegacyHashForXomlFileChecksum" to true. In code:

C#

```
System.AppContext.SetSwitch("Switch.System.Workflow.ComponentModel.UseLegacyHashForXomlFileChecksum", true);
```

Or in a configuration file (this needs to be in MSBuild.exe.config for the MSBuild.exe that you are using):

XML

```
<configuration>
<runtime>
<AppContextSwitchOverrides
value="Switch.System.Workflow.ComponentModel.UseLegacyHashForXomlFileChecksum=true" />
</runtime>
</configuration>
```

Name	Value
Scope	Minor
Version	4.8
Type	Retargeting

.NET Framework 4.8.1

No app compatibility issues were introduced in .NET Framework 4.8.1.

Mitigate new behaviors in .NET Framework 4.6 and later

Article • 09/15/2021

This section contains articles that describe mitigations for behaviors that were introduced in .NET Framework 4.6 and later.

Consider mitigating a new behavior if it causes problems for your app or is otherwise undesirable. In some cases, you can also *enable* a new behavior on apps that target an older version of .NET Framework but are running on .NET Framework 4.6 or later.

Mitigation: Custom [IMessageFilter.PreFilterMessage](#) Implementations

Article • 04/12/2022

In Windows Forms apps that target versions of the .NET Framework starting with the .NET Framework 4.6.1, a custom [IMessageFilter.PreFilterMessage](#) implementation can safely filter messages when the [Application.FilterMessage](#) method is called if the [IMessageFilter.PreFilterMessage](#) implementation:

- Does one or both of the following:
 - Adds a message filter by calling the [AddMessageFilter](#) method.
 - Removes a message filter by calling the [RemoveMessageFilter](#) method.
- And pumps messages by calling the [Application.DoEvents](#) method.

Impact

This change only affects Windows Forms apps that target versions of the .NET Framework starting with the .NET Framework 4.6.1.

For Windows Forms apps that target previous versions of the .NET Framework, such implementations in some cases throw an [IndexOutOfRangeException](#) exception when the [Application.FilterMessage](#) method is called

Mitigation

If this change is undesirable, apps that target the .NET Framework 4.6.1 or a later version can opt out of it by adding the following configuration setting to the [`<runtime>`](#) section of the app's configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Windows.Forms.DontSupportReentrantFilterMessage=true"
  />
</runtime>
```

In addition, apps that target previous versions of the .NET Framework but are running under the .NET Framework 4.6.1 or a later version can opt in to this behavior by adding the following configuration setting to the `<runtime>` section of the app's configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Windows.Forms.DontSupportReentrantFilterMessage=false"
  />
</runtime>
```

See also

- [Application compatibility](#)

Mitigation: Deserialization of Objects Across App Domains

Article • 03/11/2022

In some cases, when an app uses two or more app domains with different application bases, the attempt to deserialize objects in the logical call context across app domains throws an exception.

Diagnosing the issue

The issue arises under the following sequence of conditions:

1. An app uses two or more app domains with different application bases.
2. Some types are explicitly added to the [LogicalCallContext](#) by calling a method such as [LogicalCallContext.SetData](#) or [CallContext.LogicalSetData](#). These types are not marked as serializable and are not stored in the global assembly cache.
3. Later, code running in the non-default app domain tries to read a value from a configuration file or use XML to deserialize an object.
4. In order to read from a configuration file or deserialize the object, an [XmlReader](#) object tries to access the configuration system.
5. If the configuration system has not already been initialized, it must complete its initialization. This means, among other things, that the runtime has to create a stable path for a configuration system, which it does as follows:
 - a. It looks for evidence for the non-default app domain.
 - b. It tries to calculate the evidence for the non-default app domain based on the default app domain.
 - c. The call to get evidence for the default app domain triggers a cross-app domain call from the non-default app domain to the default app domain.
 - d. As part of the cross-app domain contract in the .NET Framework, the contents of the logical call context also have to be marshalled across app domain boundaries.
6. Because the types that are in the logical call context cannot be resolved in the default app domain, an exception is thrown.

Mitigation

To work around this issue, do the following

1. Look for the call to `get_Evidence` in the call stack when the exception is thrown.
The exception can be any of a large subset of exceptions, including [FileNotFoundException](#) and [SerializationException](#).
2. Identify the place in the app where no objects are added to the logical call context and add the following code:

C#

```
System.Configuration.ConfigurationManager.GetSection("system.xml/xmlReader");
```

See also

- [Application compatibility](#)

Mitigation: New 64-bit JIT Compiler

Article • 09/15/2021

Starting with .NET Framework 4.6, the runtime includes a new 64-bit JIT compiler for just-in-time compilation. This change does not affect compilation with the 32-bit JIT compiler.

Unexpected behavior or exceptions

In some cases, compilation with the new 64-bit JIT compiler results in a runtime exception or in behavior that is not observed when executing code compiled by the older 64-bit JIT compiler. The known differences include the following:

Important

All of these known issues have been addressed in the new 64-bit compiler released with the .NET Framework 4.6.2. Most have also been addressed in service releases of the .NET Framework 4.6 and 4.6.1 that are included with Windows Update. You can eliminate these issues by ensuring that your version of Windows is up to date, or by upgrading to the .NET Framework 4.6.2.

- Under certain conditions, an unboxing operation may throw a [NullReferenceException](#) in Release builds with optimization turned on.
- In some cases, execution of production code in a large method body may throw a [StackOverflowException](#).
- Under certain conditions, structures passed to a method are treated as reference types rather than value types in Release builds. One of the manifestations of this issue is that the individual items in a collection appear in an unexpected order.
- Under certain conditions, the comparison of [UInt16](#) values with their high bit set is incorrect if optimization is enabled.
- Under certain conditions, particularly when initializing array values, memory initialization by the [OpCodes.Initblk](#) IL instruction may initialize memory with an incorrect value. This can result either in an unhandled exception or incorrect output.
- Under certain rare conditions, a conditional bit test can return the incorrect [Boolean](#) value or throw an exception if compiler optimizations are enabled.

- Under certain conditions, if an `if` statement is used to test for a condition before entering a `try` block and in the exit from the `try` block, and the same condition is evaluated in the `catch` or `finally` block, the new 64-bit JIT compiler removes the `if` condition from the `catch` or `finally` block when it optimizes code. As a result, code inside the `if` statement in the `catch` or `finally` block is executed unconditionally.

Mitigation of known issues

If you encounter the issues listed above, you can address them by doing any of the following:

- Upgrade to the .NET Framework 4.6.2. The new 64-bit compiler included with the .NET Framework 4.6.2 addresses each of these known issues.
- Ensure that your version of Windows is up to date by running Windows Update. Service updates to the .NET Framework 4.6 and 4.6.1 address each of these issues except the [NullReferenceException](#) in an unboxing operation.
- Compile with the older 64-bit JIT compiler. See the [Mitigation of other issues](#) section for more information on how to do this.

Mitigation of other issues

If you encounter any other difference in behavior between code compiled with the older 64-bit compiler and the new 64-bit JIT compiler, or between the debug and release versions of your app that are both compiled with the new 64-bit JIT compiler, you can do the following to compile your app with the older 64-bit JIT compiler:

- On a per-application basis, you can add the `<useLegacyJit>` element to your application's configuration file. The following disables compilation with the new 64-bit JIT compiler and instead uses the legacy 64-bit JIT compiler.

XML

```
<?xml version ="1.0"?>
<configuration>
    <runtime>
        <useLegacyJit enabled="1" />
    </runtime>
</configuration>
```

- On a per-user basis, you can add a `REG_DWORD` value named `useLegacyJit` to the `HKEY_CURRENT_USER\Software\Microsoft\.NETFramework` key of the registry. A value of 1 enables the legacy 64-bit JIT compiler; a value of 0 disables it and enables the new 64-bit JIT compiler.
- On a per-machine basis, you can add a `REG_DWORD` value named `useLegacyJit` to the `HKEY_LOCAL_MACHINE\Software\Microsoft\.NETFramework` key of the registry. A value of 1 enables the legacy 64-bit JIT compiler; a value of 0 disables it and enables the new 64-bit JIT compiler.

You can also let us know about the problem by reporting a bug on [Microsoft Connect](#).

See also

- [Application compatibility](#)
- [`<useLegacyJit>` Element](#)

Mitigation: Path Colon Checks

Article • 09/15/2021

Starting with apps that target the .NET Framework 4.6.2, a number of changes were made to support previously unsupported paths (both in terms of length and format). In particular, checks for the proper drive separator syntax (the colon) were made more correct.

Impact

These changes block some URI paths the [Path.GetDirectoryName](#) and [Path.GetPathRoot](#) methods previously supported.

Mitigation

To work around the problem of a previously acceptable path that is no longer supported by the [Path.GetDirectoryName](#) and [Path.GetPathRoot](#) methods, you can do the following:

- Manually remove the scheme from a URL. For example, remove `file://` from a URL.
- Pass the URI to a [Uri](#) constructor, and retrieve the value of the [Uri.LocalPath](#) property.
- Opt out of the new path normalization by setting the `Switch.System.IO.UseLegacyPathHandling` [AppContext](#) switch to `true`.

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.UseLegacyPathHandling=true" />
</runtime>
```

See also

- [Application compatibility](#)

Mitigation: Path Normalization

Article • 09/15/2021

Starting with apps that target .NET Framework 4.6.2, path normalization in the .NET Framework has changed.

What is path normalization?

Normalizing a path involves modifying the string that identifies a path or file so that it conforms to a valid path on the target operating system. Normalization typically involves:

- Canonicalizing component and directory separators.
- Applying the current directory to a relative path.
- Evaluating the relative directory (.) or the parent directory (..) in a path.
- Trimming specified characters.

The changes

Starting with apps that target the .NET Framework 4.6.2, path normalization has changed in the following ways:

- The runtime defers to the operating system's [GetFullPathName](#) function to normalize paths.
- Normalization no longer involves trimming the end of directory segments (such as a space at the end of a directory name).
- Support for device path syntax in full trust, including \\.\ and, for file I/O APIs in mscorlib.dll, \\?\.
- The runtime does not validate device syntax paths.
- The use of device syntax to access alternate data streams is supported.

Impact

For apps that target the .NET Framework 4.6.2 or later, these changes are on by default. They should improve performance while allowing methods to access previously

inaccessible paths.

Apps that target the .NET Framework 4.6.1 and earlier versions but are running under the .NET Framework 4.6.2 or later are unaffected by this change.

Mitigation

Apps that target the .NET Framework 4.6.2 or later can opt out of this change and use legacy normalization by adding the following to the `<runtime>` section of the application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.UseLegacyPathHandling=true" />
</runtime>
```

Apps that target the .NET Framework 4.6.1 or earlier but are running on the .NET Framework 4.6.2 or later can enable the changes to path normalization by adding the following line to the `<runtime>` section of the application .configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.UseLegacyPathHandling=false" />
</runtime>
```

See also

- [Application compatibility](#)

Mitigation: PNG Frames in Icon Objects

Article • 07/23/2022

Starting with .NET Framework 4.6, the [Icon.ToBitmap](#) method successfully converts icons with PNG frames into [Bitmap](#) objects.

In apps that target the .NET Framework 4.5.2 and earlier versions, the [Icon.ToBitmap](#) method throws an [ArgumentOutOfRangeException](#) exception if the [Icon](#) object has PNG frames.

Impact

This change affects apps that are recompiled to target the .NET Framework 4.6 and that implement special handling for the [ArgumentOutOfRangeException](#) that is thrown when an [Icon](#) object has PNG frames. When running under .NET Framework 4.6, the conversion is successful, an [ArgumentOutOfRangeException](#) is no longer thrown, and therefore the exception handler is no longer invoked.

Mitigation

If this behavior is undesirable, you can retain the previous behavior by adding the following element to the [<runtime>](#) section of your app.config file:

XML

```
<AppContextSwitchOverrides  
    value="Switch.System.Drawing.DontSupportPngFramesInIcons=true" />
```

If the app.config file already contains the [AppContextSwitchOverrides](#) element, the new value should be merged with the [value](#) attribute like this:

XML

```
<AppContextSwitchOverrides  
    value="Switch.System.Drawing.DontSupportPngFramesInIcons=true;previous  
key=previous-value" />
```

See also

- Application compatibility

Mitigation: Pointer-based Touch and Stylus Support

Article • 09/15/2021

WPF applications that target the .NET Framework 4.7 and are running on Windows starting with Windows 10 Creators Update can enable an optional `WM_POINTER`-based WPF touch/stylus stack.

Impact

Developers who do not explicitly enable pointer-based touch/stylus support should see no change in WPF touch/stylus behavior.

The following are current known issues with the optional `WM_POINTER`-based touch/stylus stack:

- No support for real-time inking.

While inking and stylus plugins still work, they are processed on the UI thread, which can lead to poor performance.

- Behavioral changes due to changes in promotion from touch/stylus events to mouse events.
 - Manipulation may behave differently.
 - Drag/Drop will not show appropriate feedback for touch input. (This does not affect stylus input.)
 - Drag/Drop can no longer be initiated on touch/stylus events.

This can potentially cause the application to become unresponsive until mouse input is detected. Instead, developers should initiate drag and drop from mouse events.

Opting in to `WM_POINTER`-based touch/stylus support

Developers who wish to enable this stack can add the following to their application's `app.config` file.

XML

```
<configuration>
  <runtime>
    <AppContextSwitchOverrides
value="Switch.System.Windows.Input.Stylus.EnablePointerSupport=true"/>
  </runtime>
</configuration>
```

Removing this entry or setting its value to `false` turns this optional stack off.

See also

- [Application compatibility](#)

Mitigation: Pool Blocking Period

Article • 09/15/2021

The connection pool blocking period has been removed for connections to Azure SQL databases.

Additional description

In the .NET Framework 4.6.1 and earlier versions, when an app encounters a transient connection failure when connecting to a database, the connection attempt cannot be retried quickly, because the connection pool caches the error and re-throws it for 5 seconds to 1 min. For more information, see [SQL Server Connection Pooling \(ADO.NET\)](#). This behavior is problematic for connections to Azure SQL databases, which often fail with transient errors that are typically recovered from within a few seconds. The connection pool blocking feature means that the app cannot connect to the database for an extensive period, even though the database is available. This behavior is particularly problematic for web apps that connect to Azure SQL databases and that need to render within a few seconds.

Starting with the .NET Framework 4.6.2, for connection open requests to known Azure SQL databases (*.database.windows.net, *.database.chinacloudapi.cn, *.database.usgovcloudapi.net, *.database.cloudapi.de), connection open errors are not cached. For all other connection attempts, the connection pool blocking period continues to be enforced.

Impact

This change allows the connection open attempt to be retried immediately for Azure SQL databases, thereby improving the performance of cloud-enabled apps.

Mitigation

For apps that are adversely affected by this change, the connection pool blocking period can be configured by setting the new [SqlConnectionStringBuilder.PoolBlockingPeriod](#) property. The value of the property is a member of the [System.Data.SqlClient.PoolBlockingPeriod](#) enumeration that can take either of three values:

- [PoolBlockingPeriod.AlwaysBlock](#)

- [PoolBlockingPeriod.Auto](#)
- [PoolBlockingPeriod.NeverBlock](#)

The previous behavior can be restored by setting the [PoolBlockingPeriod](#) property to [PoolBlockingPeriod.AlwaysBlock](#).

See also

- [Application compatibility](#)

Mitigation: Product versioning

Article • 07/23/2022

In .NET Framework 4.6 and later versions, product versioning has changed from the previous releases of .NET Framework (.NET Framework 4, 4.5, 4.5.1, and 4.5.2).

Product versioning changes

The following are the detailed changes:

- The value of the `Version` entry in the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` key has changed to `4.6.xxxxx` for the .NET Framework 4.6 and its point releases, and to `4.7.xxxxx` for .NET Framework 4.7. In .NET Framework 4.5, 4.5.1, and 4.5.2, it had the format `4.5.xxxxx`.
- The file and product versioning for .NET Framework files has changed from the earlier versioning scheme of `4.0.30319.x` to `4.6.x.0` for the .NET Framework 4.6 and its point releases, and to `4.7.x.0` for the .NET Framework 4.7 and its point releases. You can see these new values when you view the file's **Properties** after right-clicking on a file.
- The [AssemblyFileVersionAttribute](#) and [AssemblyInformationalVersionAttribute](#) attributes for managed assemblies have `Version` values in the form `4.6.x.0` for the .NET Framework 4.6 and its point releases, and `4.7.x.0` for the .NET Framework 4.7.
- Starting with .NET Framework 4.6, the [Environment.Version](#) property returns the fixed version string `4.0.30319.42000`. In the .NET Framework 4, 4.5, 4.5.1, and 4.5.2, it returns version strings in the format `4.0.30319.xxxxx` where `xxxxx` is less than 42000 (for example, "4.0.30319.18010"). Note that we do not recommend application code taking any new dependency on the [Environment.Version](#) property.

Handling the product versioning changes

In general, applications should depend on the recommended techniques for detecting such things as the runtime version of the .NET Framework and the installation directory:

- To detect the runtime version of the .NET Framework, see [How to: Determine Which .NET Framework Versions Are Installed](#).

- To determine the installation path for the .NET Framework, use the value of the `InstallPath` entry in the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` key.

 **Important**

The subkey name is `NET Framework Setup`, not `.NET Framework Setup`.

- To determine the directory path to the .NET Framework common language runtime, call the [RuntimeEnvironment.GetRuntimeDirectory](#) method.
- To get the CLR version, call the [RuntimeEnvironment.GetSystemVersion](#) method. For the .NET Framework 4 and its point releases (.NET Framework 4.5, 4.5.1, 4.5.2, and .NET Framework 4.6, 4.6.1, 4.6.2, and 4.7), it returns the string `v4.0.30319`.

See also

- [Application compatibility](#)

Mitigation: Serialization of control characters with the DataContractJsonSerializer

Article • 09/15/2021

Starting with .NET Framework 4.7, the way in which control characters are serialized with the [DataContractJsonSerializer](#) has changed to conform to ECMAScript V6 and V8.

Impact

In .NET Framework 4.6.2 and earlier versions, the [DataContractJsonSerializer](#) did not serialize some special control characters, such as `\b`, `\f`, and `\t`, in a way that was compatible with the ECMAScript V6 and V8 standards.

For apps that target versions of .NET Framework starting with .NET Framework 4.7, serialization of these control characters is compatible with ECMAScript V6 and V8. The following APIs are affected:

- [WriteObject](#)

Mitigation

For apps that target versions of .NET Framework starting with .NET Framework 4.7, this behavior is enabled by default.

If this behavior is not desirable, you can opt out of this feature by adding the following line to the `<runtime>` section of the app.config or web.config file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Runtime.Serialization.DoNotUseECMAScriptV6EscapeControl
          Character=false" />
</runtime>
```

See also

- [Application compatibility](#)

Mitigation: TLS Protocols

Article • 07/23/2022

Starting with .NET Framework 4.6, the [System.Net.ServicePointManager](#) and [System.Net.Security.SslStream](#) classes are allowed to use one of the following three protocols: Tls1.0, Tls1.1, or Tls 1.2. The SSL3.0 protocol and RC4 cipher are not supported.

Impact

This change affects:

- Any app that uses SSL to talk to an HTTPS server or a socket server using any of the following types: [HttpClient](#), [HttpWebRequest](#), [FtpWebRequest](#), [SmtpClient](#), and [SslStream](#).
- Any server-side app that cannot be upgraded to support Tls1.0, Tls1.1, or Tls 1.2..

Mitigation

The recommended mitigation is to upgrade the sever-side app to Tls1.0, Tls1.1, or Tls 1.2. If this is not feasible, or if client apps are broken, the [AppContext](#) class can be used to opt out of this feature in either of two ways:

- Programmatically, by using a code snippet like the following:

C#

```
const string DisableCachingName =
@"TestSwitch.LocalAppContext.DisableCaching";
const string DontEnableSchUseStrongCryptoName =
@"Switch.System.Net.DontEnableSchUseStrongCrypto";
AppContext.SetSwitch(DisableCachingName, true);
AppContext.SetSwitch(DontEnableSchUseStrongCryptoName, true);
```

Because the [ServicePointManager](#) object is initialized only once, defining these compatibility settings must be the first thing the application does.

- By adding the following line to the [`<runtime>`](#) section of your app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.Net.DontEnableSchUseStrongCrypto=true"/>
```

Note, however, that opting out of the default behavior is not recommended, since it makes the application less secure.

See also

- [Application compatibility](#)

Mitigation: WCF Services and Certificate Authentication

Article • 09/15/2021

The .NET Framework 4.6 adds TLS 1.1 and TLS 1.2 to the WCF SSL protocol default list. When both client and server machines have the .NET Framework 4.6 or later installed, TLS 1.2 is used for negotiation.

Impact

TLS 1.2 does not support MD5 certificate authentication. As a result, if a customer uses an SSL certificate which uses MD5 for the hash algorithm, the WCF client fails to connect to the WCF service. For more information, see [Mitigation: WCF Services and Certificate Authentication](#).

Mitigation

You can work around this issue so that a WCF client can connect to a WCF server by doing any of the following:

- Update the certificate to not use the MD5 algorithm. This is the recommended solution.
- If the binding is not dynamically configured in source code, update the application's configuration file to use TLS 1.1 or an earlier version of the protocol. This allows you to continue to use a certificate with the MD5 hash algorithm.

⊗ Caution

This workaround is not recommended, since a certificate with the MD5 hash algorithm is considered insecure.

The following configuration file does this:

XML

```
<configuration>
    <system.serviceModel>
        <bindings>
            <netTcpBinding>
```

```
<binding>
    <security mode=
    "None|Transport|Message|TransportWithMessageCredential" >
        <transport
            clientCredentialType="None|Windows|Certificate"
            protectionLevel="None|Sign|EncryptAndSign"

            sslProtocols="Ssl3|Tls1|Tls11">
                </transport>
            </security>
        </binding>
    </netTcpBinding>
</bindings>
</system.serviceModel>
</configuration>
```

- If the binding is dynamically configured in source code, update the `TcpTransportSecurity.SslProtocols` property to use TLS 1.1 (`SslProtocols.Tls11`) or an earlier version of the protocol in the source code.

 **Caution**

This workaround is not recommended, since a certificate with the MD5 hash algorithm is considered insecure.

See also

- Application compatibility

Mitigation: WPF Layout

Article • 07/23/2022

The layout of WPF controls can change slightly.

Impact

As a result of this change:

- The width or height of elements may grow or shrink by at most one pixel.
- The placement of an object can move by at most one pixel.
- Centered elements can be vertically or horizontally off center by at most one pixel.

By default, this new layout is enabled only for apps that target .NET Framework 4.6.

Mitigation

Since this modification tends to eliminate clipping of the right or bottom of WPF controls at high DPLs, apps that target earlier versions of the .NET Framework but are running on the .NET Framework 4.6 can opt into this new behavior by adding the following line to the `<runtime>` section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.MS.Internal.DoNotApplyLayoutRoundingToMarginsAndBorderThickness  
s=false" />
```

Apps that target the .NET Framework 4.6 but want WPF controls to render using the previous layout algorithm can do so by adding the following line to the `<runtime>` section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.MS.Internal.DoNotApplyLayoutRoundingToMarginsAndBorderThickness  
s=true" />
```

See also

- Application compatibility

Mitigation: WPF Window Rendering

Article • 09/15/2021

In the .NET Framework 4.6 running on Windows 8 and above, the entire window is rendered without clipping when it extends outside of single display in a multi-monitor scenario.

Impact

In general, rendering an entire window across multiple monitors without clipping is the expected behavior. However, on Windows 7 and earlier versions, WPF windows are clipped when they extend beyond a single display because rendering a portion of the window on the second monitor has a significant performance impact.

The precise impact of rendering WPF windows across monitors on Windows 8 and above is not precisely quantifiable since it depends on a large number of factors. In some cases, it may still produce an undesirable impact on performance, particularly for users who run graphics-intensive applications and have windows straddling monitors. In other cases, you may simply want a consistent behavior across .NET Framework versions.

Mitigation

You can disable this change and revert to the previous behavior of clipping a WPF window when it extends beyond a single display. There are two ways to do this:

- By adding the `<EnableMultiMonitorDisplayClipping>` element to the `<appSettings>` section of your application configuration file, you can disable or enable this behavior on apps running on Windows 8 or later. For example, the following configuration section disables rendering without clipping:

XML

```
<appSettings>
  <add key="EnableMultiMonitorDisplayClipping" value="true"/>
</appSettings>
```

The `<EnableMultiMonitorDisplayClipping>` configuration setting can have either of two values:

- `true`, to enable clipping of windows to monitor bounds during rendering.

- `false`, to disable clipping of windows to monitor bounds during rendering.
- By setting the [EnableMultiMonitorDisplayClipping](#) property to `true` at app startup.

See also

- [Application compatibility](#)

Mitigation: `X509CertificateClaimSet.FindClaims` Method

Article • 09/15/2021

Starting with apps that target .NET Framework 4.6.1, the `X509CertificateClaimSet.FindClaims` method will attempt to match the `claimType` argument with all the DNS entries in its SAN field.

Impact

This change only affects apps that target versions of the .NET Framework starting with the .NET Framework 4.6.1.

For apps that target previous versions of the .NET Framework, the `X509CertificateClaimSet.FindClaims` method attempts to match the `claimType` argument only with the last DNS entry.

Mitigation

If this change is undesirable, apps that target versions of the .NET Framework starting with the .NET Framework 4.6.1 can opt out of it by adding the following configuration setting to the `<runtime>` section of the app's configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IdentityModel.DisableMultipleDNSEntriesInSANCertificate
    =true" />
</runtime>
```

In addition, apps that target previous versions of the .NET Framework but are running under the .NET Framework 4.6.1 and later versions can opt in to this behavior by adding the following configuration setting to the `<runtime>` section of the app's configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IdentityModel.DisableMultipleDNSEntriesInSANCertificate
    =false" />
</runtime>
```

See also

- [Application compatibility](#)

Mitigation: XML Schema Validation

Article • 07/23/2022

In .NET Framework 4.6, XSD schema validation detects a violation of the unique constraint if a compound key is used and one key is empty.

Impact

The impact of this change should be minimal: based on the schema specification, a schema validation error is expected if `xsd:unique` is violated by using a compound key with an empty key.

Mitigation

Whether a schema validation error is detected if a compound key has one empty key is a configurable feature:

- Starting with the apps that target .NET Framework 4.6, detection of the schema validation error is enabled by default; however, it is possible to opt out of it, so that the schema validation error will not be detected.
- In apps that run under the .NET Framework 4.6 but target the .NET Framework 4.5.2 and earlier versions, a schema validation error is not detected by default; however, it is possible to opt into it, so that the schema validation error will be detected.

This behavior can be configured by using the `AppContext` class to define the value of the `System.Xml.IgnoreEmptyKeySequences` switch. Because the switch's default value is `false` (empty key sequences are not ignored), apps that target the .NET Framework 4.6 can opt out of the behavior by using the following code to set the switch's value to `true`:

C#

```
// Ignore empty key sequences in apps that target .NET 4.6
AppContext.SetSwitch("System.Xml.IgnoreEmptyKeySequences", true);
```

For apps that target the .NET Framework 4.5.2 and earlier versions, because the switch's default value is `true` (empty key sequences are ignored), it is possible to ensure that a

compound key with an empty key does generate a schema validation error by using the following code to set the switch's value to `false`.

C#

```
// Do not ignore empty key sequences in apps that target .NET 4.5.1 and
earlier
AppContext.SetSwitch("System.Xml.IgnoreEmptyKeySequences", false);
```

See also

- [Application compatibility](#)

Mitigation: ZipArchiveEntry.FullName Path Separator

Article • 09/15/2021

Starting with apps that target the .NET Framework 4.6.1, the path separator used in the [ZipArchiveEntry.FullName](#) property has changed from the backslash ("\") used in previous versions of .NET Framework to a forward slash ("/").

[System.IO.Compression.ZipArchiveEntry](#) objects are created by calling one of the overloads of the [ZipFile.CreateDirectory](#) method.

Impact

The change brings the .NET implementation into conformity with section 4.4.17.1 of the [.ZIP File Format Specification](#) and allows .ZIP archives to be decompressed on non-Windows systems.

Decompressing a zip file created by an app that targets a previous version of .NET Framework on non-Windows operating systems such as MacOS fails to preserve the directory structure. For example, on MacOS, it creates a set of files whose file name concatenates the directory path, any backslash ("\") characters, and the filename. As a result, the directory structure of decompressed files is not preserved.

The impact of this change on .ZIP files that are decompressed on the Windows operating system by APIs in .NET Framework [System.IO](#) namespace should be minimal, since these APIs can seamlessly handle either a slash ("/") or a backslash ("\") as the path separator character.

Mitigation

If this behavior is undesirable, you can opt out of by adding a configuration setting to the `<runtime>` section of your application configuration file. The following shows both the `<runtime>` section and the opt-out switch.

XML

```
<runtime>
    <AppContextSwitchOverrides
        value="Switch.System.IO.Compression.ZipFile.UseBackslash=true" />
</runtime>
```

In addition, apps that target previous versions of .NET Framework but are running on .NET Framework 4.6.1 and later versions can opt in to this behavior by adding a configuration setting to the `<runtime>` section of the application configuration file. The following shows both the `<runtime>` section and the opt-in switch.

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.IO.Compression.ZipFile.UseBackslash=false" />
</runtime>
```

See also

- [Application compatibility](#)

Version compatibility

Article • 08/30/2022

Backward compatibility means that an app that was developed for a particular version of a platform will run on later versions of that platform. .NET Framework tries to maximize backward compatibility: Source code written for one version of .NET Framework should compile on later versions of .NET Framework, and binaries that run on one version of .NET Framework should behave identically on later versions of .NET Framework.

Version compatibility for apps

By default, an app runs on the version of .NET Framework that it was built for. If that version isn't present and the app configuration file doesn't define supported versions, a .NET Framework initialization error may occur. In this case, the attempt to run the app will fail.

To define the specific versions on which your app runs, add one or more `<supportedRuntime>` elements to your app's configuration file. Each `<supportedRuntime>` element lists a supported version of the runtime, with the first specifying the most preferred version and the last specifying the least preferred version.

XML

```
<configuration>
  <startup>
    <supportedRuntime version="v2.0.50727" />
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

For more information, see [How to: Configure an App to Support .NET Framework 4 or 4.x](#).

Version compatibility for components

An app can control the version of the .NET Framework on which it runs, but a component can't. Components and class libraries are loaded in the context of a particular app, and that's why they automatically run on the version of the .NET Framework that the app runs on.

Because of this restriction, compatibility guarantees are especially important for components. Starting with the .NET Framework 4, you can specify the degree to which a component is expected to remain compatible across multiple versions by applying the [System.Runtime.Versioning.ComponentGuaranteesAttribute](#) attribute to that component. Tools can use this attribute to detect potential violations of the compatibility guarantee in future versions of a component.

Backward compatibility

The .NET Framework 4.5 and later versions are backward-compatible with apps that were built with earlier versions of the .NET Framework. In other words, apps and components built with previous versions will work without modification on the .NET Framework 4.5 and later versions. However, by default, apps run on the version of the common language runtime for which they were developed, so you may have to provide a configuration file to enable your app to run on the .NET Framework 4.5 or later versions. For more information, see the [Version compatibility for apps](#) section earlier in this article.

In practice, this compatibility can be broken by seemingly inconsequential changes in the .NET Framework and changes in programming techniques. For example, performance improvements in the .NET Framework 4.5 can expose a race condition that did not occur on earlier versions. Similarly, using a hard-coded path to .NET Framework assemblies, performing an equality comparison with a particular version of the .NET Framework, and getting the value of a private field by using reflection are not backward-compatible practices. In addition, each version of the .NET Framework includes bug fixes and security-related changes that can affect the compatibility of some apps and components.

If your app or component doesn't work as expected on .NET Framework 4.5 or a later version, use the following checklists:

- If your app was developed to run on any version of the .NET Framework starting with .NET Framework 4.0, see [Application compatibility](#) to generate lists of changes between your targeted .NET Framework version and the version on which your app is running.
- If you have a .NET Framework 3.5 app, also see [.NET Framework 4 Migration Issues](#).
- If you have a .NET Framework 2.0 app, also see [Changes in .NET Framework 3.5 SP1](#).
- If you have a .NET Framework 1.1 app, also see [Changes in .NET Framework 2.0](#).

- If you're recompiling existing source code to run on the .NET Framework 4.5 or its point releases, or if you're developing a new version of an app or component that targets the .NET Framework 4.5 or its point releases from an existing source code base, check [What's Obsolete in the Class Library](#) for obsolete types and members, and apply the workaround described. (Previously compiled code will continue to run against types and members that have been marked as obsolete.)
- If you determine that a change in the .NET Framework 4.5 has broken your app, check the [Runtime Settings Schema](#), and particularly the [`< ApplicationContextSwitchOverrides >` Element](#), to determine whether you can use a runtime setting in your app's configuration file to restore the previous behavior.
- If you come across an issue that isn't documented, open a problem on the [Developer Community site for .NET](#) or open an issue in the [Microsoft/dotnet GitHub repo](#).

Side-by-side execution

If you can't find a suitable workaround for your issue, remember that .NET Framework 4.5 (or one of its point releases) runs side by side with versions 1.1, 2.0, and 3.5, and is an in-place update that replaces version 4. For apps that target versions 1.1, 2.0, and 3.5, you can install the appropriate version of .NET Framework on the target machine to run the app in its best environment. For more information about side-by-side execution, see [Side-by-Side Execution](#).

See also

- [What's New](#)
- [What's Obsolete in the Class Library](#)
- [Application Compatibility](#)
- [.NET Framework official support policy](#)
- [.NET Framework 4 Migration Issues](#)

.NET Framework versions and dependencies

Article • 04/24/2024

Each version of .NET Framework contains the common language runtime (CLR), the base class libraries, and other managed libraries. This article describes the key features of .NET Framework by version, provides information about the underlying CLR versions and associated development environments, and identifies the versions that are installed by the Windows operating system (OS).

Each new version of .NET Framework adds new features but retains features from previous versions.

ⓘ Note

.NET Framework [is serviced monthly](#) with security and reliability bug fixes. .NET Framework will continue to be included with Windows, with no plans to remove it. You don't need to migrate your .NET Framework apps, but for new development, use [.NET 6 or later](#).

The CLR is identified by its own version number. The .NET Framework version number is incremented at each release, but the CLR version is not always incremented. For example, .NET Framework 4, 4.5, and later releases include CLR 4, but .NET Framework 2.0, 3.0, and 3.5 include CLR 2.0. (There was no version 3 of the CLR.)

💡 Tip

- For a complete list of supported operating systems, see [System requirements](#).
- For downloads, see [Install .NET Framework for developers](#).
- For information about determining which versions of .NET Framework are installed on a computer, see [How to determine which .NET Framework versions are installed](#).

Version information

The tables that follow summarize .NET Framework version history and correlate each version with Visual Studio, Windows, and Windows Server. Visual Studio supports multi-

targeting, so you're not limited to the version of .NET Framework that's listed.

- The check mark icon denotes OS versions on which .NET Framework is installed by default.
- The plus sign icon denotes OS versions on which .NET Framework doesn't come installed but can be installed.
- The asterisk * denotes OS versions on which .NET Framework (whether preinstalled or not) must be enabled [in Control Panel](#) or, for Windows Server, through the Server Manager.

Jump to:

- [.NET Framework 4.8.1](#)
- [.NET Framework 4.8](#)
- [.NET Framework 4.7.2](#)
- [.NET Framework 4.7.1](#)
- [.NET Framework 4.7](#)
- [.NET Framework 4.6.2](#)
- [.NET Framework 4.6.1](#)
- [.NET Framework 4.6](#)
- [.NET Framework 4.5.2](#)
- [.NET Framework 4.5.1](#)
- [.NET Framework 4.5](#)
- [.NET Framework 4](#)
- [.NET Framework 3.5](#)
- [.NET Framework 3.0](#)
- [.NET Framework 2.0](#)
- [.NET Framework 1.1](#)
- [.NET Framework 1.0](#)

.NET Framework 4.8.1

- [New features](#)
- [New accessibility features](#)
- [Release notes ↗](#)

Expand table

Versions	
CLR	4

Versions	
Windows	<ul style="list-style-type: none"> ✓ October 2023 Release (Version 22631) ✓ September 2022 Release (Version 22621) + 11 October 2021 Release (Version 22000) + 10 November 2021 Update + 10 May 2021 Update + 10 October 2020 Update
Windows Server	+ Windows Server 2022

To determine the installed .NET version, use the following `Release` DWORD:

- 533320 (Windows 11 September 2022 Release and Windows 11 October 2023 Release)
- 533325 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.8

- [New features](#)
- [New in accessibility](#)
- [Release notes ↗](#)

[+] Expand table

Versions	
CLR	4
Windows	<ul style="list-style-type: none"> ✓ 11 October 2021 Release (Version 22000) ✓ 10 November 2021 Update ✓ 10 May 2021 Update ✓ 10 October 2020 Update ✓ 10 May 2020 Update ✓ 10 November 2019 Update ✓ 10 May 2019 Update + 10 October 2018 Update (Version 1809) + 10 April 2018 Update (Version 1803) + 10 Fall Creators Update (Version 1709) + 10 Creators Update (Version 1703) + 10 Anniversary Update (Version 1607) + 8.1 + 7

Versions	
Windows Server	✓ Windows Server 2022 ✚ Windows Server 2019 ✚ Windows Server, version 1809 ✚ Windows Server, version 1803 ✚ 2016 ✚ 2012 R2 ✚ 2012 ✚ 2008 R2 SP1

To determine the installed .NET version, use the following `Release` DWORD:

- 528449 (Windows 11 and Windows Server 2022)
- 528372 (Windows 10 May 2020 Update and Windows 10 October 2020 Update and Windows 10 May 2021 Update)
- 528040 (Windows 10 May 2019 Update and Windows 10 November 2019 Update)
- 528049 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.7.2

- [New features](#)
- [New in accessibility](#)
- [Release notes ↗](#)

[] [Expand table](#)

Versions	
CLR	4
Included in Visual Studio	2019†
Windows	✓ 10 October 2018 Update (Version 1809) ✓ 10 April 2018 Update (Version 1803) ✚ 10 Fall Creators Update (Version 1709) ✚ 10 Creators Update (Version 1703) ✚ 10 Anniversary Update (Version 1607) ✚ 8.1 ✚ 7
Windows Server	✓ Windows Server 2019 ✓ Windows Server, version 1809 ✓ Windows Server, version 1803 ✚ Windows Server, version 1709

Versions

- + 2016
- + 2012 R2
- + 2012
- + 2008 R2 SP1

[†]Requires installing the .NET desktop development, ASP.NET and web development, Azure development, Office/SharePoint development, Mobile development with .NET, or .NET Core cross-platform development workloads.

To determine the installed .NET version, use the following `Release` DWORD:

- 461814 (Windows 10 October 2018 Update)
- 461808 (Windows 10 April 2018 Update and Windows Server, version 1803)
- 461814 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.7.1

- [New features](#)
- [New in accessibility](#)
- [Release notes ↗](#)

[] [Expand table](#)

Versions

CLR	4
Windows	<ul style="list-style-type: none">✓ 10 Fall Creators Update (Version 1709)+ 10 Creators Update (Version 1703)+ 10 Anniversary Update (Version 1607)+ 8.1+ 7
Windows Server	<ul style="list-style-type: none">+ Windows Server, version 1803✓ Windows Server, version 1709+ 2016+ 2012 R2+ 2012+ 2008 R2 SP1

To determine the installed .NET version, use the following `Release` DWORD:

- 461308 (Windows 10 Creators Update and Windows Server, version 1709)

- 461310 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.7

- [New features](#)
- [Release notes ↗](#)

 [Expand table](#)

Versions	
CLR	4
Windows	 10 Creators Update (Version 1703)  10 Anniversary Update (Version 1607)  8.1  7
Windows Server	 2016  2012 R2  2012  2008 R2 SP1

To determine the installed .NET version, use the following `Release` DWORD:

- 460798 (Windows 10 Creators Update)
- 460805 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.6.2

- [New features](#)
- [Release notes ↗](#)

 [Expand table](#)

Versions	
CLR	4
Windows	 10 Anniversary Update (Version 1607)  10 November Update (Version 1511)  10

Versions	
	8.1
	7
Windows Server	2016
	2012 R2
	2012
	2008 R2 SP1
	2008 SP2

To determine the installed .NET version, use the following `Release` DWORD:

- 394802 (Windows 10 Anniversary Update and Windows Server 2016)
- 394806 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.6.1

- [New features](#)
- [Release notes ↗](#)

[\[\]](#) Expand table

Versions	
CLR	4
Included in Visual Studio	2017 ¹
Windows	10 November Update (Version 1511) 10 8.1 8 7
Windows Server	2012 R2 2012 2008 R2 SP1

¹ Requires installing the .NET desktop development, ASP.NET and web development, Azure development, Office/SharePoint development, Mobile development with .NET, or .NET Core cross-platform development workloads.

To determine the installed .NET version, use the following `Release` DWORD:

- 394254 (Windows 10 November Update)
- 394271 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.6

- [New features](#)
- [Release notes ↗](#)

[] [Expand table](#)

Versions	
CLR	4
Included in Visual Studio	2015
Windows	10 8.1 8 7 Vista
Windows Server	2012 R2 2012 2008 R2 SP1 2008 SP2

To determine the installed .NET version, use the following `Release` DWORD:

- 393295 (Windows 10)
- 393297 (all other OS versions)

For more information, see [instructions](#).

.NET Framework 4.5.2

- [New features](#)
- [Release notes ↗](#)

[] [Expand table](#)

Versions	
CLR	4
Windows	8.1 8 7 Vista
Windows Server	2012 R2 2012 2008 R2 SP1 2008 SP2

To determine the installed .NET version, use `Release` DWORD 379893. For more information, see [instructions](#).

.NET Framework 4.5.1

- [New features](#)
- [Release notes ↗](#)

Expand table

Versions	
CLR	4
Included in Visual Studio	2013
Windows	8.1 8 7 Vista
Windows Server	2012 R2 2012 2008 R2 SP1 2008 SP2

To determine the installed .NET version, use the following `Release` DWORD:

- 378675 (Windows 8.1)
- 378758 (all other Windows versions)

For more information, see [instructions](#).

Important

Starting with Visual Studio 2022, Visual Studio no longer includes .NET Framework components for .NET Framework 4.0 - 4.5.1 because these versions are no longer supported. Visual Studio 2022 and later versions can't build apps that target .NET Framework 4.0 through .NET Framework 4.5.1. To continue building these apps, you can use Visual Studio 2019 or an earlier version.

.NET Framework 4.5

- [New features](#)
- [Release notes](#) ↗

 [Expand table](#)

Versions	
CLR	4
Included in Visual Studio	2012
Windows	 8  7  Vista
Windows Server	 2012  2008 R2 SP1  2008 SP2

To determine the installed .NET version, use `Release` DWORD 378389. For more information, see [instructions](#).

Important

Starting with Visual Studio 2022, Visual Studio no longer includes .NET Framework components for .NET Framework 4.0 - 4.5.1 because these versions are no longer supported. Visual Studio 2022 and later versions can't build apps that target .NET Framework 4.0 through .NET Framework 4.5.1. To continue building these apps, you can use Visual Studio 2019 or an earlier version.

.NET Framework 4

New features

[+] Expand table

Versions	
CLR	4
Included in Visual Studio	2010
Windows	 7  Vista
Windows Server	 2008 R2 SP1  2008 SP2  2003

To determine installed .NET version: See [instructions](#).

Important

Starting with Visual Studio 2022, Visual Studio no longer includes .NET Framework components for .NET Framework 4.0 - 4.5.1 because these versions are no longer supported. Visual Studio 2022 and later versions can't build apps that target .NET Framework 4.0 through .NET Framework 4.5.1. To continue building these apps, you can use Visual Studio 2019 or an earlier version.

.NET Framework 3.5

New features:

- LINQ
- Expression trees
- Improved ASP.NET support for AJAX development
- HashSet collections
- DateTimeOffset
- WCF and WF integration
- Peer-to-Peer networking
- Add-ins for extensibility

[+] Expand table

Versions	
CLR	2.0
Included in Visual Studio	2008
Windows	✓ 10* ✓ 8.1* ✓ 8* ✓ 7 + Vista
Windows Server	+ Windows Server, version 1803* + Windows Server, version 1709* + 2016* + 2012 R2* + 2012* ✓ 2008 R2 SP1* + 2008 SP2 + 2003

To determine installed .NET version: See [instructions](#).

.NET Framework 3.0

New features:

- Windows Presentation Foundation
- Windows Communication Foundation
- Windows Workflow Foundation
- Windows CardSpace

[] Expand table

Versions	
CLR	2.0
Windows	✓ Vista
Windows Server	✓ 2008 R2 SP1* ✓ 2008 SP2*
	+ 2003

To determine installed .NET version: See [instructions](#).

.NET Framework 2.0

New features:

- Generics
- Debugger edit and continue
- Improved scalability and performance
- ClickOnce deployment
- In ASP.NET 2.0, new controls and support for a broad array of browsers
- 64-bit support

[] [Expand table](#)

	Versions
CLR	2.0
Included in Visual Studio	2005
Windows	N/A
Windows Server	✓ 2008 R2 SP1 ✓ 2008 SP2 ✓ 2003

To determine installed .NET version: See [instructions](#).

.NET Framework 1.1

New features:

- ASP.NET mobile controls
- Side-by-side execution
- IPv6 support

[] [Expand table](#)

	Versions
CLR	1.1
Included in Visual Studio	2003

	Versions
Windows	N/A
Windows Server	✓ 2003

To determine installed .NET version: See [instructions](#).

.NET Framework 1.0

[\[+\] Expand table](#)

	Versions
CLR	1.0
Included in Visual Studio	Visual Studio .NET
Windows	N/A
Windows Server	N/A

To determine installed .NET version: See [instructions](#).

ⓘ Note

- .NET Framework must be enabled on this operating system through [Control Panel \(for Windows\) or the Server Manager \(for Windows Server\)](#).
- In general, you should not uninstall any versions of .NET Framework that are installed on your computer, because an application you use may depend on a specific version and may break if that version is removed. You can load multiple versions of .NET Framework on a single computer at the same time. This means that you can install .NET Framework without having to uninstall previous versions. For more information, see [Getting Started](#).

Remarks for version 4.5 and later

.NET Framework 4.5 is an in-place update that replaces .NET Framework 4 on your computer, and similarly, .NET Framework 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, and 4.8 are in-place updates to .NET Framework 4.5. In-place update means that they use the same runtime version, but the assembly versions are updated and include new types and members. After you install one of these updates, your .NET Framework 4, .NET

Framework 4.5, .NET Framework 4.6, or .NET Framework 4.7 apps should continue to run without requiring recompilation. However, the reverse is not true. We do not recommend running apps that target a later version of .NET Framework on an earlier version. For example, we do not recommend that you run an app that targets .NET Framework 4.6 on .NET Framework 4.5.

The following guidelines apply:

- In Visual Studio, you can choose .NET Framework 4.5 as the target framework for a project (this sets the [GetReferenceAssemblyPaths.TargetFrameworkMoniker](#) property) to compile the project as a .NET Framework 4.5 assembly or executable. This assembly or executable can then be used on any computer that has .NET Framework 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, or 4.8 installed.
- In Visual Studio, you can choose .NET Framework 4.5.1 as the target framework for a project to compile it as a .NET Framework 4.5.1 assembly or executable. Only run this assembly or executable on computers that have .NET Framework 4.5.1 or later installed. An executable that targets .NET Framework 4.5.1 will be blocked from running on a computer that only has an earlier version of .NET Framework, such as .NET Framework 4.5, installed. The user will be prompted to install .NET Framework 4.5.1. In addition, .NET Framework 4.5.1 assemblies should not be called from an app that targets an earlier version of .NET Framework, such as .NET Framework 4.5.

 **Note**

.NET Framework 4.5.1 and .NET Framework 4.5 are used here only as examples. The principle described applies to any app that targets a later version of .NET Framework than the one installed on the system on which it's running.

Some changes in .NET Framework may require changes to your app code; see [Application Compatibility](#) before you run your existing apps with .NET Framework 4.5 or later versions. For more information about installing the current version, see [Install the .NET Framework for developers](#). For information about support for the .NET Framework, see [.NET Framework official support policy](#) on the .NET website.

Remarks for older versions

.NET Framework versions 2.0, 3.0, and 3.5 are built with the same version of the CLR (CLR 2.0). These versions represent successive layers of a single installation. Each version is built incrementally on top of the earlier versions. It's not possible to run versions 2.0, 3.0,

and 3.5 side by side on a computer. When you install version 3.5, you get the 2.0 and 3.0 layers automatically, and apps that were built for versions 2.0, 3.0, and 3.5 can all run on version 3.5. However, .NET Framework 4 ends this layering approach, and it and later releases (.NET Framework 4.5, 4.5.1, 4.5.2, 4.6, 4.6.1, 4.6.2, 4.7, 4.7.1, 4.7.2, and 4.8) also represent successive layers of a single installation. Starting with .NET Framework 4, you can use in-process, side by side hosting to run multiple versions of the CLR in a single process. For more information, see [Assemblies and Side-by-Side Execution](#).

In addition, if your app targets version 2.0, 3.0, or 3.5, your users may be required to enable .NET Framework 3.5 on a Windows 8, Windows 8.1, or Windows 10 computer before they can run your app. For more information, see [Install the .NET Framework 3.5 on Windows 11, Windows 10, Windows 8.1, and Windows 8](#).

Important

Starting with Visual Studio 2022, Visual Studio no longer includes .NET Framework components for .NET Framework 4.0 - 4.5.1 because these versions are no longer supported. Visual Studio 2022 and later versions can't build apps that target .NET Framework 4.0 through .NET Framework 4.5.1. To continue building these apps, you can use Visual Studio 2019 or an earlier version.

Next steps

- If you're new to the .NET Framework, see the [overview](#) for an introduction to key concepts and features.
- For new features and improvements in the .NET Framework 4.5 and its point releases, see [What's new in the .NET Framework](#).
- For information about migrating your app to a newer version of the .NET Framework, see the [migration guide](#).
- For information about determining which versions or updates are installed on a computer, see [How to: Determine Which .NET Framework Versions Are Installed](#) and [How to: Determine Which .NET Framework Updates Are Installed](#).

See also

- [Version compatibility](#)
- [.NET Framework official support policy ↗](#)
- [Troubleshoot blocked .NET Framework installations and uninstallations](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

How to: Determine which .NET Framework versions are installed

Article • 04/05/2024

ⓘ Note

This article is specific to .NET Framework. To determine which .NET Core and .NET 5+ SDKs and runtimes are installed, see [How to check that .NET is already installed](#).

You can install and run multiple versions of .NET Framework on a computer.

- If you want to check the versions on your *own* computer, the easiest way is through **Control Panel > Programs > Programs and Features**, or in **Settings** under **Apps > Installed apps**. You can also use [these community-maintained tools](#).
- If you're an app developer, you might need to know which .NET Framework versions are installed on the app user's computer. The [registry](#) contains a list of the versions of .NET Framework installed on the computer. You can also query the [RuntimeInformation.FrameworkDescription property](#).
- To find the CLR version, which is versioned separately, see [Find CLR versions](#).

.NET Framework consists of two main components, which are versioned separately:

- A set of assemblies, which are collections of types and resources that provide the functionality for your apps. .NET Framework and the assemblies share the same version number. For example, .NET Framework versions include 4.5, 4.6.1, and 4.7.2.
- The common language runtime (CLR), which manages and executes your app's code. A single CLR version typically supports multiple .NET Framework versions. For example, CLR version 4.0.30319.xxxxx where xxxx is less than 42000, supports .NET Framework versions 4 through 4.5.2. CLR version greater than or equal to 4.0.30319.42000 supports .NET Framework versions starting with .NET Framework 4.6.

💡 Tip

For information about detecting the installed *updates* for each version of .NET Framework, see [How to: Determine which .NET Framework updates are installed](#).

Community-maintained tools

Community-maintained tools are available to help detect which .NET Framework versions are installed:

- [https://github.com/jmalarcon/DotNetVersions ↗](https://github.com/jmalarcon/DotNetVersions)
- [https://github.com/EliteLoser/DotNetVersionLister ↗](https://github.com/EliteLoser/DotNetVersionLister)

RuntimelInformation.FrameworkDescription property

To programmatically query for which .NET version your app is running on, you can use the [RuntimelInformation.FrameworkDescription](#) property. If the app is running on .NET Framework, the output will be similar to:

Output

```
.NET Framework 4.8.4250.0
```

By comparison, if the app is running on .NET Core or .NET 5+, the output will be similar to:

Output

```
.NET Core 3.1.9  
.NET 5.0.0
```

Registry

You can use the registry to detect which .NET Framework version is installed. The keys are different for .NET Framework 1.0-4.0 and .NET Framework 4.5+. You can use Registry Editor, PowerShell, or code to check the registry.

- [.NET Framework 4.5 and later versions](#)
 - [Registry Editor](#)
 - [Query using code](#)
 - [Query using PowerShell](#)
- [.NET Framework 1.0-4.0](#)
 - [Registry Editor](#)
 - [Query using code](#)
 - [Query using PowerShell](#)

.NET Framework 4.5 and later versions

The version of .NET Framework (4.5 and later) installed on a machine is listed in the registry at **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full**. If the **Full** subkey is missing, then .NET Framework 4.5 or above isn't installed.

 **Note**

The **NET Framework Setup** subkey in the registry path does *not* begin with a period.

The **Release REG_DWORD** value in the registry represents the version of .NET Framework installed.

 Expand table

.NET Framework version	Value of Release
.NET Framework 4.5	All Windows operating systems: 378389
.NET Framework 4.5.1	On Windows 8.1 and Windows Server 2012 R2: 378675 On all other Windows operating systems: 378758
.NET Framework 4.5.2	All Windows operating systems: 379893
.NET Framework 4.6	On Windows 10: 393295 On all other Windows operating systems: 393297
.NET Framework 4.6.1	On Windows 10 November Update systems: 394254 On all other Windows operating systems (including Windows 10): 394271
.NET Framework 4.6.2	On Windows 10 Anniversary Update and Windows Server 2016: 394802 On all other Windows operating systems (including other Windows 10 operating systems): 394806
.NET Framework 4.7	On Windows 10 Creators Update: 460798 On all other Windows operating systems (including other Windows 10 operating systems): 460805
.NET Framework 4.7.1	On Windows 10 Fall Creators Update and Windows Server, version 1709: 461308 On all other Windows operating systems (including other Windows 10 operating systems): 461310

.NET Framework version	Value of Release
.NET Framework 4.7.2	On Windows 10 April 2018 Update and Windows Server, version 1803: 461808 On all Windows operating systems other than Windows 10 April 2018 Update and Windows Server, version 1803: 461814
.NET Framework 4.8	On Windows 10 May 2019 Update and Windows 10 November 2019 Update: 528040 On Windows 10 May 2020 Update, October 2020 Update, May 2021 Update, November 2021 Update, and 2022 Update: 528372 On Windows 11 and Windows Server 2022: 528449 On all other Windows operating systems (including other Windows 10 operating systems): 528049
.NET Framework 4.8.1	On Windows 11 2022 Update and Windows 11 2023 Update: 533320 All other Windows operating systems: 533325

Minimum version

To determine whether a *minimum* version of .NET Framework is present, check for a **Release** REG_DWORD value that's greater than or equal to the corresponding value listed in the following table. For example, if your application runs under .NET Framework 4.8 or a later version, test for a **Release** REG_DWORD value that's *greater than or equal* to 528040.

[\[+\] Expand table](#)

.NET Framework version	Minimum value
.NET Framework 4.5	378389
.NET Framework 4.5.1	378675
.NET Framework 4.5.2	379893
.NET Framework 4.6	393295
.NET Framework 4.6.1	394254
.NET Framework 4.6.2	394802
.NET Framework 4.7	460798
.NET Framework 4.7.1	461308
.NET Framework 4.7.2	461808

.NET Framework version	Minimum value
.NET Framework 4.8	528040
.NET Framework 4.8.1	533320

Use Registry Editor

1. From the **Start** menu, choose **Run**, enter *regedit*, and then select **OK**.

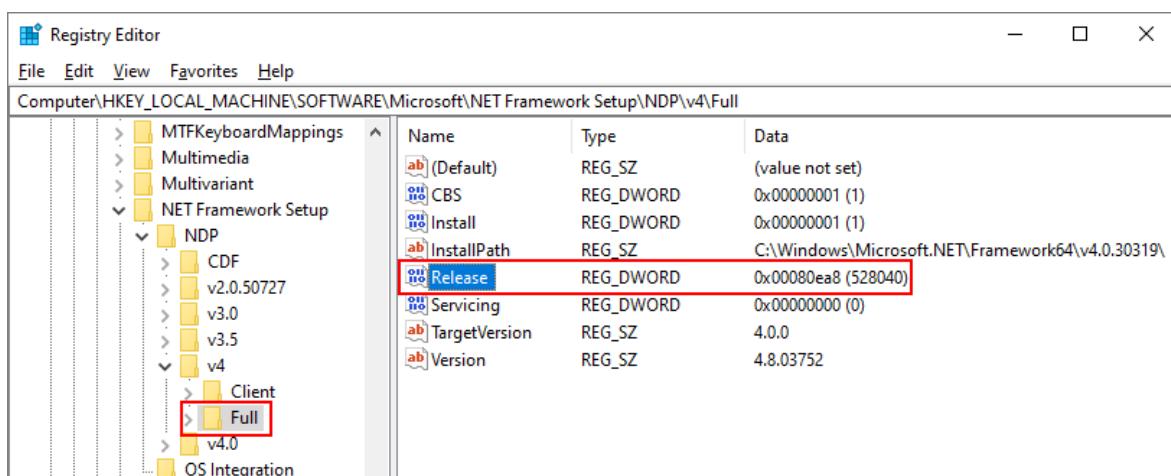
(You must have administrative credentials to run regedit.)

2. In the Registry Editor, open the following subkey:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework

Setup\NDP\v4\Full. If the **Full** subkey isn't present, then you don't have .NET Framework 4.5 or later installed.

3. Check for a REG_DWORD entry named **Release**. If it exists, then you have .NET Framework 4.5 or later installed. Its value corresponds to a particular version of .NET Framework. In the following figure, for example, the value of the **Release** entry is 528040, which is the release key for .NET Framework 4.8.



Use PowerShell to check for a minimum version

Use PowerShell commands to check the value of the **Release** entry of the **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full** subkey.

The following examples check the value of the **Release** entry to determine whether .NET Framework 4.6.2 or later is installed. This code returns `True` if it's installed and `False` otherwise.

PowerShell

```
(Get-ItemPropertyValue -LiteralPath 'HKLM:SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full' -Name Release) -ge 394802
```

Query the registry using code

1. Use the [RegistryKey.OpenBaseKey](#) and [RegistryKey.OpenSubKey](#) methods to access the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full` subkey in the Windows registry.

Important

If the app you're running is 32-bit and running in 64-bit Windows, the registry paths will be different than previously listed. The 64-bit registry is available in the `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\` subkey. For example, the registry subkey for .NET Framework 4.5 is `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\NET Framework Setup\NDP\v4\Full`.

2. Check the `Release` REG_DWORD value to determine the installed version. To be forward-compatible, check for a value greater than or equal to the value listed in the [.NET Framework version table](#).

The following example checks the value of the `Release` entry in the registry to find the versions of .NET Framework 4.5-4.8.1 that are installed.

Tip

Add the directive `using Microsoft.Win32` or `Imports Microsoft.Win32` at the top of your code file if you haven't already done so.

C#

```
const string subkey = @"SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full\";

using (var ndpKey = RegistryKey.OpenBaseKey(RegistryHive.LocalMachine,
RegistryView.Registry32).OpenSubKey(subkey))
{
    if (ndpKey != null && ndpKey.GetValue("Release") != null)
    {
        Console.WriteLine($"".NET Framework Version:
```

```

{CheckFor45PlusVersion((int)ndpKey.GetValue("Release"))}" );
}
else
{
    Console.WriteLine(".NET Framework Version 4.5 or later is not
detected.");
}
}

// Checking the version using >= enables forward compatibility.
string CheckFor45PlusVersion(int releaseKey)
{
    if (releaseKey >= 533320)
        return "4.8.1 or later";
    if (releaseKey >= 528040)
        return "4.8";
    if (releaseKey >= 461808)
        return "4.7.2";
    if (releaseKey >= 461308)
        return "4.7.1";
    if (releaseKey >= 460798)
        return "4.7";
    if (releaseKey >= 394802)
        return "4.6.2";
    if (releaseKey >= 394254)
        return "4.6.1";
    if (releaseKey >= 393295)
        return "4.6";
    if (releaseKey >= 379893)
        return "4.5.2";
    if (releaseKey >= 378675)
        return "4.5.1";
    if (releaseKey >= 378389)
        return "4.5";
    // This code should never execute. A non-null release key should mean
    // that 4.5 or later is installed.
    return "No 4.5 or later version detected";
}

```

The example displays output like the following:

Output

.NET Framework Version: 4.6.1

Query the registry using PowerShell

The following example uses PowerShell to check the value of the **Release** entry in the registry to find the versions of .NET Framework 4.5-4.8.1 that are installed:

PowerShell

```
$release = Get-ItemPropertyValue -LiteralPath 'HKLM:SOFTWARE\Microsoft\.NET Framework Setup\NDP\v4\Full' -Name Release
switch ($release) {
    { $_ -ge 533320 } { $version = '4.8.1 or later'; break }
    { $_ -ge 528040 } { $version = '4.8'; break }
    { $_ -ge 461808 } { $version = '4.7.2'; break }
    { $_ -ge 461308 } { $version = '4.7.1'; break }
    { $_ -ge 460798 } { $version = '4.7'; break }
    { $_ -ge 394802 } { $version = '4.6.2'; break }
    { $_ -ge 394254 } { $version = '4.6.1'; break }
    { $_ -ge 393295 } { $version = '4.6'; break }
    { $_ -ge 379893 } { $version = '4.5.2'; break }
    { $_ -ge 378675 } { $version = '4.5.1'; break }
    { $_ -ge 378389 } { $version = '4.5'; break }
    default { $version = $null; break }
}

if ($version) {
    Write-Host -Object ".NET Framework Version: $version"
} else {
    Write-Host -Object '.NET Framework Version 4.5 or later is not detected.'
}
```

This example follows the recommended practice for version checking:

- It checks whether the value of the **Release** entry is *greater than or equal to* the value of the known release keys.
- It checks in order from most recent version to earliest version.

.NET Framework 1.0-4.0

Each version of .NET Framework from 1.1 to 4.0 is listed as a subkey at **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP**. The following table lists the path to each .NET Framework version. For most versions, there's an **Install REG_DWORD** value of **1** to indicate this version is installed. In these subkeys, there's also a **Version REG_SZ** value that contains a version string.

ⓘ Note

The **NET Framework Setup** subkey in the registry path does *not* begin with a period.

[Expand table](#)

Framework Version	Registry Subkey	Value
1.0	HKLM\Software\Microsoft\.NETFramework\Policy\v1.0\3705	Install REG_SZ equals 1
1.1	HKLM\Software\Microsoft\NET Framework Setup\NDP\v1.1.4322	Install REG_DWORD equals 1
2.0	HKLM\Software\Microsoft\NET Framework Setup\NDP\v2.0.50727	Install REG_DWORD equals 1
3.0	HKLM\Software\Microsoft\NET Framework Setup\NDP\v3.0\Setup	InstallSuccess REG_DWORD equals 1
3.5	HKLM\Software\Microsoft\NET Framework Setup\NDP\v3.5	Install REG_DWORD equals 1
4.0 Client Profile	HKLM\Software\Microsoft\NET Framework Setup\NDP\v4\Client	Install REG_DWORD equals 1
4.0 Full Profile	HKLM\Software\Microsoft\NET Framework Setup\NDP\v4\Full	Install REG_DWORD equals 1

ⓘ Important

If the app you're running is 32-bit and running in 64-bit Windows, the registry paths will be different than previously listed. The 64-bit registry is available in the HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ subkey. For example, the registry subkey for .NET Framework 3.5 is HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\NET Framework Setup\NDP\v3.5.

Notice that the registry path to the .NET Framework 1.0 subkey is different from the others.

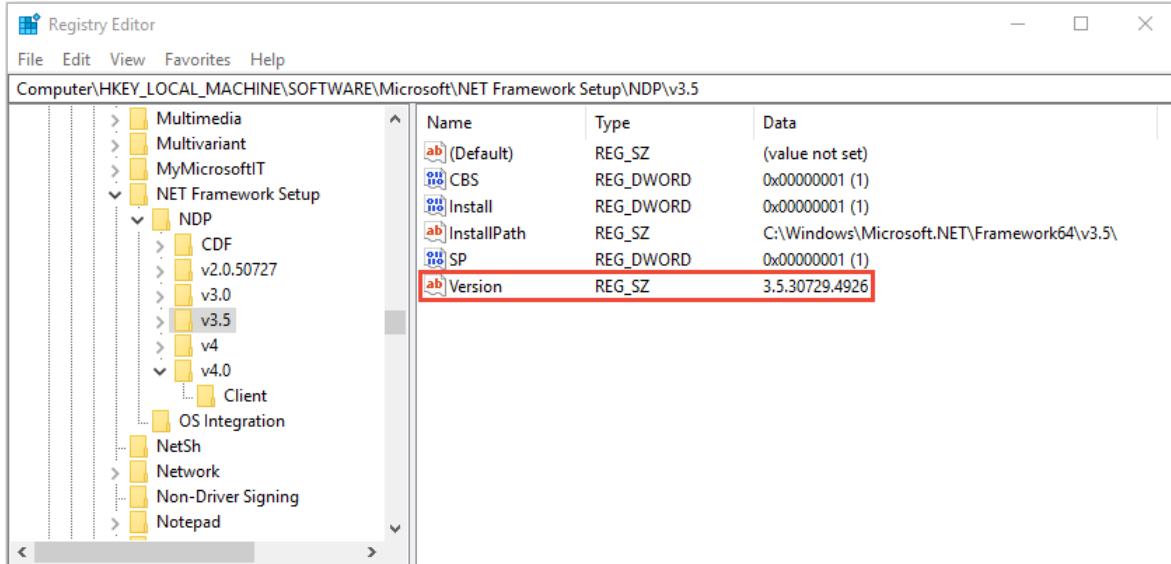
Use Registry Editor (older framework versions)

1. From the Start menu, choose Run, enter *regedit*, and then select OK.

You must have administrative credentials to run regedit.

2. Open the subkey that matches the version you want to check. Use the table in the [.NET Framework 1.0-4.0](#) section.

The following figure shows the subkey and its **Version** value for .NET Framework 3.5.



Query the registry using code (older framework versions)

Use the [Microsoft.Win32.RegistryKey](#) class to access the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP` subkey in the Windows registry.

ⓘ Important

If the app you're running is 32-bit and running in 64-bit Windows, the registry paths will be different than previously listed. The 64-bit registry is available in the `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\` subkey. For example, the registry subkey for .NET Framework 3.5 is

`HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\NET Framework Setup\NDP\v3.5.`

The following example finds the versions of .NET Framework 1-4 that are installed:

```
C#
```

```
// Open the registry key for the .NET Framework entry.
using (RegistryKey ndpKey =
    RegistryKey.OpenBaseKey(RegistryHive.LocalMachine,
RegistryView.Registry32).
    OpenSubKey(@"SOFTWARE\Microsoft\NET Framework Setup\NDP\")) {
    foreach (var versionKeyName in ndpKey.GetSubKeyNames())
    {
        // Skip .NET Framework 4.5 version information.
        if (versionKeyName == "v4")
        {
            continue;
        }

        if (versionKeyName.StartsWith("v"))
        {
            RegistryKey versionKey = ndpKey.OpenSubKey(versionKeyName);

            // Get the .NET Framework version value.
            var name = (string)versionKey.GetValue("Version", "");
            // Get the service pack (SP) number.
            var sp = versionKey.GetValue("SP", "").ToString();

            // Get the installation flag.
            var install = versionKey.GetValue("Install", "").ToString();
            if (string.IsNullOrEmpty(install))
            {
                // No install info; it must be in a child subkey.
                Console.WriteLine($"{versionKeyName} {name}");
            }
            else if (install == "1")
            {
                // Install = 1 means the version is installed.

                if (!string.IsNullOrEmpty(sp))
                {
                    Console.WriteLine($"{versionKeyName} {name} SP{sp}");
                }
                else
                {
                    Console.WriteLine($"{versionKeyName} {name}");
                }
            }
        }

        if (!string.IsNullOrEmpty(name))
        {
            continue;
        }
        // else print out info from subkeys...

        // Iterate through the subkeys of the version subkey.
        foreach (var subKeyName in versionKey.GetSubKeyNames())
        {
            RegistryKey subKey = versionKey.OpenSubKey(subKeyName);
```

```
name = (string)subKey.GetValue("Version", "");
if (!string.IsNullOrEmpty(name))
    sp = subKey.GetValue("SP", "").ToString();

install = subKey.GetValue("Install", "").ToString();
if (string.IsNullOrEmpty(install))
{
    // No install info; it must be later.
    Console.WriteLine($"  {versionKeyName}  {name}");
}
else if (install == "1")
{
    if (!string.IsNullOrEmpty(sp))
    {
        Console.WriteLine($"  {subKeyName}  {name}
SP{sp}");
    }
    else
    {
        Console.WriteLine($"  {subKeyName}  {name}");
    }
}
}
}
}
```

The example displays output similar to the following:

Output

v2.0.50727 2.0.50727.4927 SP2
v3.0 3.0.30729.4926 SP2
v3.5 3.5.30729.4926 SP1
v4.0
Client 4.0.0.0

Query the registry using PowerShell (older framework versions)

The following example uses PowerShell to check the value of the **Release** entry in the registry to find the versions of .NET Framework 1-4 that are installed:

PowerShell

```
Get-ChildItem -Path 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP' |  
Where-Object { ($_.PSChildName -ne "v4") -and ($_.PSChildName -like 'v*') }  
|  
ForEach-Object {  
    $name = $_.Version  
    $sp = $_.SP
```

```

$install = $_.Install
if (-not $install) {
    Write-Host -Object "$($_.PSChildName)  $($name)"
}
elseif ($install -eq '1') {
    if (-not $sp) {
        Write-Host -Object "$($_.PSChildName)  $($name)"
    }
    else {
        Write-Host -Object "$($_.PSChildName)  $($name) SP $($sp)"
    }
}
if (-not $name) {
    $parentName = $_.PSChildName
    Get-ChildItem -LiteralPath $_.PSPath |
    Where-Object {
        if ($.Property -contains 'Version') { $name = Get-
ItemPropertyValue -Path $_.PSPath -Name Version }
        if ($name -and ($.Property -contains 'SP')) { $sp = Get-
ItemPropertyValue -Path $_.PSPath -Name SP }
        if ($.Property -contains 'Install') { $install = Get-
ItemPropertyValue -Path $_.PSPath -Name Install }
        if (-not $install) {
            Write-Host -Object "  $($parentName)  $($name)"
        }
        elseif ($install -eq '1') {
            if (-not $sp) {
                Write-Host -Object "  $($_.PSChildName)  $($name)"
            }
            else {
                Write-Host -Object "  $($_.PSChildName)  $($name)
SP $($sp)"
            }
        }
    }
}
}
}

```

Find CLR versions

The .NET Framework CLR installed with .NET Framework is versioned separately. There are two ways to detect the version of the .NET Framework CLR:

- The Clrver.exe tool
- The Environment.Version property

The Clrver.exe tool

Use the [CLR Version tool \(Clrver.exe\)](#) to determine which versions of the CLR are installed on a computer. Open [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#) and enter `clrver`.

Sample output:

```
Console

Versions installed on the machine:
v2.0.50727
v4.0.30319
```

The `Environment.Version` property

Important

For .NET Framework 4.5 and later versions, don't use the [Environment.Version](#) property to detect the version of the CLR. Instead, query the registry as described in [.NET Framework 4.5 and later versions](#).

1. Query the [Environment.Version](#) property to retrieve a `Version` object.

The returned `System.Version` object identifies the version of the runtime that's currently executing the code. It doesn't return assembly versions or other versions of the runtime that may have been installed on the computer.

For .NET Framework versions 4, 4.5, 4.5.1, and 4.5.2, the string representation of the returned `Version` object has the form 4.0.30319.xxxxx, where xxxx is less than 42000. For .NET Framework 4.6 and later versions, it has the form 4.0.30319.42000.

2. After you have the `Version` object, query it as follows:

- For the major release identifier (for example, *4* for version 4.0), use the [Version.Major](#) property.
- For the minor release identifier (for example, *0* for version 4.0), use the [Version.Minor](#) property.
- For the entire version string (for example, *4.0.30319.18010*), use the [Version.ToString](#) method. This method returns a single value that reflects the version of the runtime that's executing the code. It doesn't return assembly versions or other runtime versions that may be installed on the computer.

The following example uses the [Environment.Version](#) property to retrieve CLR version information:

C#

```
Console.WriteLine($"Version: {Environment.Version}");
```

The example displays output similar to the following:

Output

```
Version: 4.0.30319.18010
```

See also

- [How to: Determine which .NET Framework updates are installed](#)
- [Troubleshoot: Determine which versions and service packs of .NET Framework are installed](#)
- [Install .NET Framework for developers](#)
- [.NET Framework versions and dependencies](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

How to determine which .NET Framework security updates and hotfixes are installed

Article • 04/05/2024

This article shows you how to find out which .NET Framework security updates and hotfixes are installed on a computer.

Update history

To see which .NET Framework updates are installed on your own computer, in **Settings**, navigate to **Windows Update > Update history**. Look under the **Quality Updates** section for .NET Framework updates. For example, you might see an update similar to "2023-11 Cumulative Update for .NET Framework 3.5 and 4.8.1 for Windows 11, version 22H2 for x64 (KB5032007)".

Registry

You can query the registry using [Registry Editor](#), [code](#), or [PowerShell](#).

ⓘ Note

All the registry techniques require an account with administrative privileges.

Use Registry Editor

The installed security updates and hotfixes for each version of the .NET Framework installed on a computer are listed in the Windows registry. You can use the Registry Editor (`regedit.exe`) program to view this information.

1. Open the program `regedit.exe`. In Windows 8 and later versions, right-click **Start** , then select **Run**. In the **Open** box, enter `regedit` and select **OK**.
2. In the Registry Editor, open the following subkey:

`HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Updates`

The installed updates are listed under subkeys that identify the .NET Framework version they apply to. Each update is identified by a Knowledge Base (KB) number.

In the Registry Editor, the .NET Framework versions and installed updates for each version are stored in different subkeys. For information about detecting the installed version numbers, see [How to: Determine which .NET Framework versions are installed](#).

Query using code

The following example programmatically determines the .NET Framework security updates and hotfixes that are installed on a computer:

```
C#  
  
using System;  
using Microsoft.Win32;  
  
public class GetUpdateHistory  
{  
    public static void Main()  
    {  
        using (RegistryKey baseKey =  
RegistryKey.OpenBaseKey(RegistryHive.LocalMachine,  
RegistryView.Registry32).OpenSubKey(@"SOFTWARE\Microsoft\Updates"))  
        {  
            foreach (string baseKeyName in baseKey.GetSubKeyNames())  
            {  
                if (baseKeyName.Contains(".NET Framework"))  
                {  
                    using (RegistryKey updateKey =  
baseKey.OpenSubKey(baseKeyName))  
                    {  
                        Console.WriteLine(baseKeyName);  
                        foreach (string kbKeyName in  
updateKey.GetSubKeyNames())  
                        {  
                            using (RegistryKey kbKey =  
updateKey.OpenSubKey(kbKeyName))  
                            {  
                                Console.WriteLine(" " + kbKeyName);  
                            }  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

The example produces an output that's similar to the following one:

Console

```
Microsoft .NET Framework 4 Client Profile
KB2468871
KB2468871v2
KB2478063
KB2533523
KB2544514
KB2600211
KB2600217
Microsoft .NET Framework 4 Extended
KB2468871
KB2468871v2
KB2478063
KB2533523
KB2544514
KB2600211
KB2600217
```

Query using PowerShell

The following example shows how to determine the .NET Framework security updates and hotfixes that are installed on a computer using PowerShell:

PowerShell

```
$DotNetVersions = Get-ChildItem HKLM:\SOFTWARE\WOW6432Node\Microsoft\Updates
| Where-Object {$_.name -like
"*.NET Framework*"}

ForEach($Version in $DotNetVersions){

    $Updates = Get-ChildItem $Version.PSPPath
    $Version.PSChildName
    ForEach ($Update in $Updates){
        $Update.PSChildName
    }
}
```

The example produces an output that's similar to the following one:

Console

```
Microsoft .NET Framework 4 Client Profile
KB2468871
KB2468871v2
KB2478063
KB2533523
KB2544514
```

KB2600211
KB2600217
Microsoft .NET Framework 4 Extended
KB2468871
KB2468871v2
KB2478063
KB2533523
KB2544514
KB2600211
KB2600217

See also

- [How to: Determine which .NET Framework versions are installed](#)
- [Install the .NET Framework for developers](#)
- [Versions and dependencies](#)

 [Collaborate with us on GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

How to: Configure an App to Support .NET Framework 4 or later versions

Article • 07/23/2022

All apps that host the common language runtime (CLR) need to start, or *activate*, the CLR in order to run managed code. Typically, a .NET Framework app runs on the version of the CLR that it was built on, but you can change this behavior for desktop apps by using an application configuration file (sometimes referred to as an `app.config` file). However, you cannot change the default activation behavior for Windows Store apps or Windows Phone apps by using an application configuration file. This article explains how to enable your desktop app to run on another version of the .NET Framework and provides an example of how to target version 4 or later versions.

The version of the .NET Framework that an app runs on is determined in the following order:

- Configuration file.

If the application configuration file includes `<supportedRuntime>` entries that specify one or more .NET Framework versions, and one of those versions is present on the user's computer, the app runs on that version. The configuration file reads `<supportedRuntime>` entries in the order they are listed, and uses the first .NET Framework version listed that is present on the user's computer. (Use the `<requiredRuntime> element` for version 1.0.)

- Compiled version.

If there is no configuration file, but the version of the .NET Framework that the app was built on is present on the user's computer, the app runs on that version.

- Latest version installed.

If the version of the .NET Framework that the app was built on is not present and a configuration file does not specify a version in a `<supportedRuntime> element`, the app tries to run on the latest version of the .NET Framework that is present on the user's computer.

However, .NET Framework 1.0, 1.1, 2.0, 3.0, and 3.5 apps do not automatically run on the .NET Framework 4 or later, and in some cases, the user may receive an error and may be prompted to install .NET Framework 3.5. The activation behavior may also depend on the user's operating system, because different versions of Windows system include different versions of the .NET Framework. If your app

supports both the .NET Framework 3.5 and 4 or later, we recommend that you indicate this with multiple entries in the configuration file to avoid .NET Framework initialization errors. For more information, see [Versions and Dependencies](#).

You might also want to configure your .NET Framework 3.5 apps to run on the .NET Framework 4 or later versions, even on computers that have the .NET Framework 3.5 installed, to take advantage of the performance improvements in versions 4 and later versions.

Important

We recommend that you always test your app on every .NET Framework version that you support. See [Version Compatibility](#) for information about upgrading your application to support later .NET Framework versions.

For information about modifying your .NET Framework 1.0 and 1.1 apps to support Windows 7 and Windows 8, see [Migrating from the .NET Framework 1.1](#).

To configure your app to run on the .NET Framework 4 or later versions

1. Add or locate the configuration file for the .NET Framework project. The configuration file for an app is in the same directory and has the same name as the app, but has a .config extension. For example, for an app named MyExecutable.exe, the application configuration file is named MyExecutable.exe.config.

To add a configuration file, on the Visual Studio menu bar, choose **Project, Add New Item**. Choose **General** from the left pane, and then choose **Configuration File**. Name the configuration file *App.config*. These menu choices are not available for Windows Store app or Windows phone app projects, because you cannot change the activation policy on those platforms.

2. Add the `<supportedRuntime>` element as follows to the application configuration file:

XML

```
<configuration>
  <startup>
    <supportedRuntime version="version"/>
  </startup>
</configuration>
```

where *<version>* specifies the CLR version that aligns with the .NET Framework version that your app supports. Use the following strings:

- .NET Framework 1.0: "v1.0.3705"
- .NET Framework 1.1: "v1.1.4322"
- .NET Framework 2.0, 3.0, and 3.5: "v2.0.50727"
- .NET Framework 4 and later versions: "v4.0"

You can add multiple `<supportedRuntime>` elements, listed in order of preference, to specify support for multiple versions of the .NET Framework.

The following table demonstrates how application configuration file settings and the .NET Framework versions installed on a computer determine the version that a .NET Framework 3.5 app runs on. The examples are specific to a .NET Framework 3.5 application, but you can use similar logic to target applications built with earlier .NET Framework versions. Note that the .NET Framework 2.0 version number (v2.0.50727) is used to specify the .NET Framework 3.5 in the application configuration file.

App.config file setting	On computer with version 3.5 installed	On computer with versions 3.5 and 4 or later versions installed	On computer with version 4 or later versions installed
None	Runs on 3.5	Runs on 3.5	Displays error message that prompts user to install the correct version*
<code><supportedRuntime version="v2.0.50727"/></code>	Runs on 3.5	Runs on 3.5	Displays error message that prompts user to install the correct version*
<code><supportedRuntime version="v2.0.50727"/></code> <code><supportedRuntime version="v4.0"/></code>	Runs on 3.5	Runs on 3.5	Runs on 4 or later versions
<code><supportedRuntime version="v4.0"/></code> <code><supportedRuntime version="v2.0.50727"/></code>	Runs on 3.5	Runs on 4 or later versions	Runs on 4 or later versions

App.config file setting	On computer with version 3.5 installed	On computer with versions 3.5 and 4 or later versions installed	On computer with version 4 or later versions installed
<supportedRuntime version="v4.0"/>	Displays error message that prompts user to install the correct version*	Runs on 4 or later versions	Runs on 4 or later versions

* For more information about this error message and ways to avoid it, see [.NET Framework Initialization Errors: Managing the User Experience](#).

See also

- [Migrating from the .NET Framework 1.1](#)
- [Migration Guide](#)

Migrate from the .NET Framework 1.1

Article • 07/23/2022

Windows 7 and later versions of the Windows operating system do not support .NET Framework 1.1. As a result, applications that target the .NET Framework 1.1 will not run without modification on Windows 7 or later operating system versions. This topic discusses the steps required to run an application that targets the .NET Framework 1.1 under Windows 7 and later versions of the Windows operating system. For more information about the .NET Framework 1.1 and Windows 8, see [Run .NET Framework 1.1 Apps on Windows 8 and later versions](#).

Retarget or recompile

There are two ways to get an application that was compiled using the .NET Framework 1.1 to run on Windows 7 or a later Windows operating system:

- Retarget the application to run under .NET Framework 4 and later versions. Retargeting requires that you add a `<supportedRuntime>` element to the application's configuration file that allows it to run under .NET Framework 4 and later versions. Such a configuration file takes the following form:

XML

```
<configuration>
  <startup>
    <supportedRuntime version="v4.0"/>
  </startup>
</configuration>
```

- Recompile the application with a compiler that targets the .NET Framework 4 or a later version. If you originally used Visual Studio 2003 to develop and compile your solution, you can open the solution in Visual Studio 2010 (and possibly later versions too) and use the **Project Compatibility** dialog box to convert the solution and project files from the formats used by Visual Studio 2003 to the Microsoft Build Engine (MSBuild) format.

Regardless of whether you prefer to recompile or retarget your application, you must determine whether your application is affected by any changes introduced in later versions of the .NET Framework. These changes are of two kinds:

- Breaking changes that occurred between the .NET Framework 1.1 and later versions of the .NET Framework.

- Types and type members that have been marked as deprecated or obsolete between the .NET Framework 1.1 and later versions of the .NET Framework.

Whether you retarget your application or recompile it, you should review both the breaking changes and the obsolete types and members for each version of the .NET Framework that was released after .NET Framework 1.1.

Breaking changes

When a breaking change occurs, depending on the specific change, a workaround may be available both for retargeted and recompiled applications. In some cases, you can add a child element to the `<runtime>` element of your application's configuration file to restore the previous behavior. For example, the following configuration file restores the string sorting and comparison behavior used in the .NET Framework 1.1 and can be used either with a retargeted or a recompiled application.

XML

```
<configuration>
  <runtime>
    <CompatSortNLSVersion enabled="4096"/>
  </runtime>
</configuration>
```

However, in some cases, you may have to modify your source code and recompile your application.

To assess the impact of possible breaking changes on your application, you must review the following lists of changes:

- [Breaking Changes in .NET Framework 2.0](#) documents changes in .NET Framework 2.0 SP1 that can affect an application that targets .NET Framework 1.1.
- [Changes in .NET Framework 3.5 SP1](#) documents changes between the .NET Framework 3.5 and the .NET Framework 3.5 SP1.
- [.NET Framework 4 Migration Issues](#) documents changes between the .NET Framework 3.5 SP1 and the .NET Framework 4.

Obsolete types and members

The impact of deprecated types and members is somewhat different for retargeted applications and recompiled applications. The use of obsolete types and members will

not affect a retargeted application unless the obsolete type or member has been physically removed from its assembly. Recompiling an application that uses obsolete types or members usually produces a compiler warning rather than a compiler error. However, in some cases, it produces a compiler error, and code that uses the obsolete type or member does not compile successfully. In this case, you must rewrite the source code that calls the obsolete type or member before you recompile your application. For more information about obsolete types and members, see [What's Obsolete in the Class Library](#).

To assess the impact of types and members that have been deprecated since the release of the .NET Framework 2.0 SP1, see [What's Obsolete in the Class Library](#). Review the lists of obsolete types and member for the .NET Framework 2.0 SP1, .NET Framework 3.5, and the .NET Framework 4.

.NET Framework 4 migration issues

Article • 03/30/2023

This article describes migration issues between .NET Framework version 3.5 Service Pack 1 and .NET Framework version 4, including fixes, changes for standards compliance and security, and changes based on customer feedback. Most of these changes do not require any programming modifications in your applications. For those that may involve modifications, see the **Recommended changes** column of the table. Notable changes are broken down by area, for example, ASP.NET and Windows Presentation Foundation (WPF).

For a higher-level overview of the issues in this article, see [Migration Guide to .NET Framework 4](#).

For information about new features, see [What's New in .NET Framework 4](#).

ASP.NET and Web

Namespaces: [System.Web](#), [System.Web.Mobile](#), [System.Web.Security](#), [System.Web.UI.WebControls](#)

Assembly: System.Web (in System.Web.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Browser definition files	<p>The browser definition files have been updated to include information about new and updated browsers and devices. Older browsers and devices such as Netscape Navigator have been removed, and newer browsers and devices such as Google Chrome and Apple iPhone have been added.</p> <p>If your application contains custom browser definitions that inherit from one of the browser definitions that have been removed, you will see an error.</p> <p>The HttpBrowserCapabilities object (which is exposed by the page's <code>Request.Browser</code> property) is driven by the browser definition files. Therefore, the information that is returned by accessing a property of this object in ASP.NET 4 might be different than the information that was returned in an earlier version of ASP.NET.</p>	<p>If your application relies on the old browser definition files, you can copy them from the following folder:</p> <p><code>Windows\Microsoft.NET\Framework\v2.0.50727\CONFIG\Browsers</code></p> <p>Copy the files into the corresponding <code>\CONFIG\Browsers</code> folder for ASP.NET 4. After you copy the files, run the Aspnet_regbrowsers.exe command-line tool. For more information, see the https://www.asp.net/mobile Web site.</p>
Child applications running under mixed versions of ASP.NET	ASP.NET 4 applications that are configured as children of applications that run earlier versions of ASP.NET might fail to start because of configuration or compilation errors. The specific error that occurs depends on whether the application runs under IIS 6.0, or under IIS 7 or IIS 7.5.	You can make changes to the configuration files of the affected applications so that the configuration system correctly recognizes the ASP.NET 4 application. For information about the changes you must make, see the section "ASP.NET 4 Child Applications Fail to Start When Under ASP.NET 2.0 or ASP.NET 3.5 Applications" in the document ASP.NET 4 Breaking Changes on the ASP.NET Web site.

Feature	Differences from 3.5 SP1	Recommended changes
ClientID changes	<p>The new <code>clientIDMode</code> setting in ASP.NET 4 lets you specify how ASP.NET generates the <code>id</code> attribute for HTML elements. In previous versions of ASP.NET, the default behavior was equivalent to the <code>AutoID</code> setting of <code>clientIDMode</code>. The default setting is now <code>Predictable</code>. For more information, see ASP.NET Web Server Control Identification.</p>	<p>If you use Visual Studio to upgrade your application from ASP.NET 2.0 or ASP.NET 3.5, the tool automatically adds a setting to the Web.config file that preserves the behavior of earlier versions of .NET Framework. However, if you upgrade an application by changing the application pool in IIS to target .NET Framework 4, ASP.NET uses the new mode by default. To disable the new client ID mode, add the following setting to the Web.config file:</p> <pre data-bbox="837 467 1187 496"><pages clientIDMode="AutoID" /></pre>
Code access security (CAS)	<p>ASP.NET 2.0 NET features that were added in ASP.NET 3.5 use the .NET Framework 1.1 and .NET Framework 2.0 code access security (CAS) model. However, the implementation of CAS in ASP.NET 4 has been substantially overhauled. As a result, partial-trust ASP.NET applications that rely on trusted code running in the global assembly cache might fail with various security exceptions. Partial-trust applications that rely on extensive modifications to machine CAS policy might also fail and throw security exceptions.</p>	<p>You can revert partial-trust ASP.NET 4 applications to the behavior of ASP.NET 1.1 and 2.0 by using the new <code>legacyCasModel</code> attribute in the <code>trust</code> configuration element, as shown in the following example:</p> <pre data-bbox="837 698 1350 727"><trust level= "Medium" legacyCasModel="true" /></pre> <p>Important: Reverting to the older CAS model might represent reduced security.</p> <p>For more information about the new ASP.NET 4 code access security model, see Code Access Security in ASP.NET 4 Applications.</p>
Configuration files	<p>The root configuration files (the <code>machine.config</code> file and the root <code>Web.config</code> file) for .NET Framework and ASP.NET 4 have been updated to include most of the boilerplate configuration information that was found in the application <code>Web.config</code> files in ASP.NET 3.5. Because of the complexity of the managed IIS 7 and IIS 7.5 configuration systems, running ASP.NET 3.5 applications under ASP.NET 4 and under IIS 7 and IIS 7.5 can result in either ASP.NET errors or IIS errors.</p>	<p>Upgrade ASP.NET 3.5 applications to ASP.NET 4 by using the project upgrade tools in Visual Studio. Visual Studio 2010 automatically modifies the ASP.NET 3.5 application's <code>Web.config</code> file to contain the appropriate settings for ASP.NET 4.</p> <p>However, you can run ASP.NET 3.5 applications using .NET Framework 4 without recompilation. In that case, you might have to manually modify the application's <code>Web.config</code> file before you run the application under .NET Framework 4 and under IIS 7 or IIS 7.5. The specific change you must make depends on the combination of software you are working with, including Service Pack (SP) releases. For information about the possible software combinations that are affected by this change and how to resolve problems with specific combinations, see the section "Configuration Errors Related to New ASP.NET 4 Root Configuration" in the document ASP.NET 4 Breaking Changes on the ASP.NET Web site.</p>

Feature	Differences from 3.5 SP1	Recommended changes
Control rendering	<p>In previous versions of ASP.NET, some controls emitted markup that you could not disable. By default, this type of markup is no longer generated in ASP.NET 4. The rendering changes affect the following controls:</p> <ul style="list-style-type: none"> * The <code>Image</code> and <code>ImageButton</code> controls no longer render a <code>border="0"</code> attribute. * The <code>BaseValidator</code> class and validation controls that derive from it no longer render red text by default. * The <code>HtmlForm</code> control does not render a <code>name</code> attribute. * The <code>Table</code> control no longer renders a <code>border="0"</code> attribute. <p>Controls that are not designed for user input (for example, the <code>Label</code> control) no longer render the <code>disabled="disabled"</code> attribute if their <code>Enabled</code> property is set to <code>false</code> (or if they inherit this setting from a container control).</p>	<p>If you use Visual Studio to upgrade your application from ASP.NET 2.0 or ASP.NET 3.5, the tool automatically adds a setting to the Web.config file that preserves legacy rendering. However, if you upgrade an application by changing the application pool in IIS to target .NET Framework 4, ASP.NET uses the new rendering mode by default. To disable the new rendering mode, add the following setting to the Web.config file:</p> <pre><pages controlRenderingCompatibilityVersion="3.5" /></pre>
Event handlers in default documents	<p>ASP.NET 4 renders the HTML <code>form</code> element's <code>action</code> attribute value as an empty string when a request is made to an extensionless URL that has a default document mapped to it. In earlier releases of ASP.NET, a request to <code>http://contoso.com</code> would result in a request to Default.aspx. In that document, the opening <code>form</code> tag would be rendered as in the following example:</p> <pre><form action="Default.aspx" /></pre> <p>In ASP.NET 4, a request to <code>http://contoso.com</code> also results in a request to Default.aspx, but ASP.NET now renders the HTML opening <code>form</code> tag as in the following example:</p> <pre><form action="" /></pre> <p>When the <code>action</code> attribute is an empty string, the IIS <code>DefaultDocumentModule</code> object creates a child request to Default.aspx. Under most conditions, this child request is transparent to application code, and the Default.aspx page runs normally. However, a potential interaction between managed code and IIS 7 or IIS 7.5 Integrated mode can cause managed .aspx pages to stop working properly during the child request. If the following conditions occur, the child request to a default .aspx document will result in an error or in unexpected behavior:</p> <p>* An .aspx page is sent to the browser with the <code>form</code> element's <code>action</code> attribute set to "".</p>	<p>For information about ways to work around problems that might arise as a result of this change, see "Event Handlers Might Not Be Raised in a Default Document in IIS 7 or IIS 7.5 Integrated Mode" in the document ASP.NET 4 Breaking Changes on the ASP.NET Web site.</p>

Feature	Differences from 3.5 SP1	Recommended changes
	<ul style="list-style-type: none"> * The form is posted back to ASP.NET. * A managed HTTP module reads some part of the entity body, such as <code>Request.Form</code> or <code>Request.Params</code>. This causes the entity body of the POST request to be read into managed memory. As a result, the entity body is no longer available to any native code modules that are running in IIS 7 or IIS 7.5 Integrated mode. * The IIS <code>DefaultDocumentModule</code> object eventually runs and creates a child request to the Default.aspx document. However, because the entity body has already been read by a piece of managed code, there is no entity body available to send to the child request. * When the HTTP pipeline runs for the child request, the handler for .aspx files runs during the handler-execute phase. <p>Because there is no entity body, there are no form variables and no view state. Therefore there is no information available for the .aspx page handler to determine what event (if any) should be raised. As a result, none of the postback event handlers for the affected .aspx page run.</p>	
Hashing algorithm	<p>ASP.NET uses both encryption and hashing algorithms to help secure data such as forms authentication cookies and view state. By default, ASP.NET 4 uses the HMACSHA256 algorithm for hash operations on cookies and view state. Earlier versions of ASP.NET used the older HMACSHA1 algorithm.</p>	<p>If you run applications that mix ASP.NET 2.0 and ASP.NET 4, where data such as forms authentication cookies must work across .NET Framework versions, configure an ASP.NET 4 Web application to use the older HMACSHA1 algorithm by adding the following setting in the Web.config file:</p> <pre data-bbox="842 1275 1191 1304"><machineKey validation="SHA1" /></pre>
Hosting controls in Internet Explorer	<p>You can no longer host Windows Forms controls in Internet Explorer, because there are better solutions for hosting controls on the Web. Therefore, the IEHost.dll and IEEExec.exe assemblies have been removed from .NET Framework.</p>	<p>You can use the following technologies for custom control development in Web applications:</p> <ul style="list-style-type: none"> * You can create a Silverlight application and configure it to run outside the browser. For more information, see Out-of-Browser Support. * You can build a XAML browser application (XBAP) to take advantage of WPF capabilities (requires .NET Framework on client machines). For more information, see WPF XAML Browser Applications Overview.
HtmlEncode and UrlEncode methods	<p>The <code>HtmlEncode</code> and <code>UrlEncode</code> methods of the <code>HttpUtility</code> and <code>HttpServerUtility</code> classes have been updated to encode the single quotation mark character (') as follows:</p> <ul style="list-style-type: none"> * The <code>HtmlEncode</code> method encodes instances of the single quotation mark as <code>&#39;</code>; * The <code>UrlEncode</code> method encodes instances of the single quotation mark as <code>%27</code> 	<p>Examine your code for places where you use the <code>HtmlEncode</code> and <code>UrlEncode</code> methods, and make sure that the change in encoding does not result in a change that would affect your application.</p>

Feature	Differences from 3.5 SP1	Recommended changes
HttpException errors in ASP.NET 2.0 applications	<p>After ASP.NET 4 has been enabled on IIS 6, ASP.NET 2.0 applications that run on IIS 6 (in either Windows Server 2003 or Windows Server 2003 R2) might generate errors such as the following: <code>System.Web.HttpException: Path '/[yourApplicationRoot]/eurl.axd/[Value]' was not found.</code></p>	<ul style="list-style-type: none"> * If ASP.NET 4 is not required in order to run the Web site, remap the site to use ASP.NET 2.0 instead. -or- * If ASP.NET 4 is required in order to run the Web site, move any child ASP.NET 2.0 virtual directories to a different Web site that is mapped to ASP.NET 2.0. -or- * Disable extensionless URLs. For more information, see "ASP.NET 2.0 Applications Might Generate <code>HttpException</code> Errors That Reference <code>eurl.axd</code>" in the document ASP.NET 4 Breaking Changes on the ASP.NET Web site.
Membership types	<p>Some types (for example, MembershipProvider) that are used in ASP.NET membership have been moved from <code>System.Web.dll</code> to the <code>System.Web.ApplicationServices.dll</code> assembly. The types were moved in order to resolve architectural layering dependencies between types in the client and in extended .NET Framework SKUs.</p>	<p>Class libraries that have been upgraded from earlier versions of ASP.NET and that use membership types that have been moved might fail to compile when used in an ASP.NET 4 project. If so, add a reference in the class library project to <code>System.Web.ApplicationServices.dll</code>.</p>
Menu control changes	<p>Changes to the <code>Menu</code> control result in the following behavior:</p> <ul style="list-style-type: none"> * If <code>MenuRenderingMode</code> is set to <code>List</code>, or if <code>MenuRenderingMode</code> is set to <code>Default</code> and <code>ControlRenderingCompatibilityVersion</code> is set to <code>4.0</code> or later, the <code>PopOutImageUrl</code> property has no effect. * If the path that is set in the <code>StaticPopOutImageUrl</code> and <code>DynamicPopOutImageUrl</code> properties contains a backslash (\), the images do not render. (In earlier versions of ASP.NET, the path could include a backslash.) 	<ul style="list-style-type: none"> * Instead of setting the <code>PopOutImageUrl</code> property for individual menu items, set the <code>StaticPopOutImageUrl</code> or <code>DynamicPopOutImageUrl</code> of the parent <code>Menu</code> control. -or- Set <code>MenuRenderingMode</code> to <code>Table</code>, or set <code>MenuRenderingMode</code> to <code>Default</code> and set <code>ControlRenderingCompatibilityVersion</code> to <code>3.5</code>. These settings cause the <code>Menu</code> control to use the HTML table-based layout that it used in earlier versions of ASP.NET. * If the path in the <code>StaticPopOutImageUrl</code> or <code>DynamicPopOutImageUrl</code> property contains a backslash (\), substitute a slash character (/).
Mobile assembly in Web.config file	<p>In previous versions of ASP.NET, a reference to the <code>System.Web.Mobile.dll</code> assembly was included in the root <code>Web.config</code> file in the <code>assemblies</code> section under <code>system.web/compilation</code>. To improve performance, the reference to this assembly has been removed.</p> <p>Note: The <code>System.Web.Mobile.dll</code> assembly and the ASP.NET mobile controls are included in ASP.NET 4, but they are deprecated.</p>	<p>If you want to use types from this assembly, add a reference to the assembly in either the root <code>Web.config</code> file or in an application <code>Web.config</code> file.</p>

Feature	Differences from 3.5 SP1	Recommended changes
Output caching	<p>In ASP.NET 1.0, a bug caused cached pages that specified <code>Location="ServerAndClient"</code> as an output cache setting to emit a <code>Vary:*</code> HTTP header in the response. This had the effect of telling client browsers to never cache the page locally. In ASP.NET 1.1, the <code>SetOmitVaryStar</code> method was added, which could be called in order to suppress the <code>Vary:*</code> header. However, bug reports suggest that developers are unaware of the existing <code>SetOmitVaryStar</code> behavior.</p> <p>In ASP.NET 4, the <code>Vary:*</code> HTTP header is no longer emitted from responses that specify the following directive:</p> <pre data-bbox="339 714 798 774"><%@ OutputCache Location="ServerAndClient" %></pre> <p>As a result, the <code>SetOmitVaryStar</code> method is no longer needed in order to suppress the <code>Vary:*</code> header. In applications that specify "ServerAndClient" for the <code>Location</code> attribute, pages will be cacheable in the browser without requiring that you call <code>SetOmitVaryStar</code>.</p>	<p>If pages in the application must emit <code>Vary:*</code>, call the <code>AppendHeader</code> method as shown in the following example:</p> <pre data-bbox="842 265 1371 287">System.Web.HttpResponse.AppendHeader("Vary", "*");</pre> <p>Alternatively, you can change the value of the output caching <code>Location</code> attribute to "Server".</p>
Page parsing	<p>The page parser for ASP.NET Web pages (.aspx files) and user controls (.ascx files) is stricter in ASP.NET 4 than in earlier versions of ASP.NET, and it flags more markup as invalid than in earlier versions.</p>	<p>Examine error messages that are produced when a page runs and fix errors that result from invalid markup.</p>
Passport types	<p>The Passport support built into ASP.NET 2.0 is obsolete and is unsupported due to changes in Passport (now Live ID SDK). As a result, the types related to Passport in <code>System.Web.Security</code> are now marked with the <code>ObsoleteAttribute</code> attribute.</p>	<p>Change any code that uses Passport types in the <code>System.Web.Security</code> namespace (for example, <code>PassportIdentity</code>) to use the Windows Live ID SDK.</p>

Feature	Differences from 3.5 SP1	Recommended changes
PathInfo information in the FilePath property	<p>ASP.NET 4 no longer includes the <code>PathInfo</code> value in the return values from properties such as <code>FilePath</code>, <code>AppRelativeCurrentExecutionFilePath</code>, and <code>CurrentExecutionFilePath</code>. Instead, the <code>PathInfo</code> information is available in <code>PathInfo</code>.</p> <p>For example, imagine the following URL fragment:</p> <pre>/testapp/Action.mvc/SomeAction</pre> <p>In earlier versions of ASP.NET, <code>HttpRequest</code> properties have the following values:</p> <ul style="list-style-type: none"> * <code>FilePath</code>: /testapp/Action.mvc/SomeAction * <code>PathInfo</code>: (empty) <p>In ASP.NET 4, <code>HttpRequest</code> properties instead have the following values:</p> <ul style="list-style-type: none"> * <code>FilePath</code>: /testapp/Action.mvc * <code>PathInfo</code>: SomeAction 	<p>Examine your code for places where you rely on properties of the <code>HttpRequest</code> class to return path information; change the code to reflect the changes in how path information is returned.</p>
Request validation	<p>To improve request validation, the ASP.NET request validation is invoked earlier in the request life cycle. As a result, request validation runs for requests that are not for .aspx files, such as for Web service calls and for custom handlers. Request validation will also be active when custom HTTP modules are running in the request processing pipeline.</p> <p>As a result of this change, requests for resources other than .aspx files might throw request validation errors. Custom code that runs in the request pipeline (for example, custom HTTP modules) might also throw request validation errors.</p>	<p>If necessary, you can revert to the old behavior of having only .aspx pages triggering request validation by using the following setting in the Web configuration file:</p> <pre><httpRuntime requestValidationMode="2.0" /></pre> <p>Warning: If you revert to the old behavior, make sure that all code in existing handlers, modules, and other custom code performs checks for potentially unsafe HTTP inputs that could be XSS attack vectors.</p>
Routing	<p>If you create a file system Web site in Visual Studio 2010 and the Web site is in a folder that contains a dot (.) in the folder name, URL routing will not work reliably. An HTTP 404 error is returned from some virtual paths. This occurs because Visual Studio 2010 launches the Visual Studio Development Server using an incorrect path for the root virtual directory.</p>	<ul style="list-style-type: none"> * In the Properties page for the file-based Web site, change the Virtual Path attribute to "/". -or- * Create a Web application project instead of a Web site project. Web application projects do not have this issue, and URL routing works even if the project folder has a dot in its name. -or- * Create an HTTP-based Web site that is hosted in IIS. IIS-hosted Web sites can have dots in the virtual path as well as in the project file folder.

Feature	Differences from 3.5 SP1	Recommended changes
SharePoint sites	If you try to run an ASP.NET 4 Web site that is deployed as a child of a SharePoint Web site that contains a custom partial-trust level named <code>WSS_Minimal</code> , you will see the following error: <pre>Could not find permission set named 'ASP.Net'.</pre>	Currently, no versions of SharePoint are compatible with ASP.NET. As a result, you should not attempt to run an ASP.NET 4 Web site as a child of a SharePoint Web site.
XHTML 1.1 standards	To enable XHTML 1.1 compliance for new Web sites, the ASP.NET controls in .NET Framework 4 will generate XHTML 1.1 compliant HTML. This rendering is enabled using the following option in the Web.config file inside the <code><system.Web></code> element: <pre><pages controlRenderingCompatibilityVersion="4.0"/></pre> This option is set by default to 4.0. Web projects that are upgraded from Visual Studio 2008 have the 3.5 setting enabled for compatibility.	None.

Core

General features

Feature	Differences from 3.5 SP1	Recommended changes
CardSpace	Windows CardSpace is no longer included in .NET Framework; it is provided separately.	Download Windows CardSpace from the Microsoft Download Center .
Configuration files	Corrections have been made in how .NET Framework accesses application configuration files.	If your application configuration file is named <i>application-name.config</i> , rename it to <i>application-name.exe.config</i> . For example, rename <i>MyApp.config</i> to <i>MyApp.exe.config</i> .
C# code compiler	The <code>Compiler</code> , <code>CompilerError</code> , and <code>ErrorLevel</code> classes that were in the <code>Microsoft.CSharp</code> namespace are no longer available, and their assembly (<code>cscompmgd.dll</code>) is no longer included in .NET Framework.	Use the <code>CodeDomProvider</code> class and other classes in the <code>System.CodeDom.Compiler</code> namespace. For more information, see Using the CodeDOM .

Feature	Differences from 3.5 SP1	Recommended changes
Hosting (unmanaged API)	To improve hosting capabilities, some of the hosting activation APIs have been deprecated. In-process side-by-side execution features enable an application to load and start multiple versions of .NET Framework in the same process. For example, you can run applications that load add-ins (or components) that are based on .NET Framework 2.0 SP1 and add-ins that are based on .NET Framework 4 in the same process. Older components continue to use the older .NET Framework version, and new components use the new .NET Framework version.	Use the configurations described in In-Process Side-by-Side Execution .
New security model	The code access security (CAS) policy has been turned off and replaced with a simplified model, as described in Security Changes in .NET Framework 4 .	Modifications may be required if you depend on CAS in your applications. For more information, see Code Access Security Policy Compatibility and Migration .

Date and time

Namespace: [System](#)

Assembly: mscorlib (in mscorlib.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Daylight savings	To be consistent with the system clock, time properties (such as Local and Now) now use operating system rules instead of other .NET Framework data for daylight saving time operations.	None.
Formatting strings	To support culture-sensitive formatting, the TimeSpan structure includes new overloads of the ToString , Parse , and TryParse methods in addition to new ParseExact and TryParseExact methods.	None.

Globalization

For a list of new neutral and specific cultures, see [What's New in Globalization and Localization](#).

Namespace: [System.Globalization](#)

Assembly: mscorlib (in mscorlib.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Culture names	The following name changes affect the German, Divehi, and African cultures: * CurrencyEnglishName : The currency name for the German (Switzerland) (de-CH) culture has changed from "sFr." to "Fr." * LongDatePattern : The long date pattern for the Divehi (Maldives) (dv-MV) culture has changed from "dd/MMMM/yyyy" to "dd/MM/yyyy". * PMDesignator : The P.M. designator of the Afrikaans (South Africa) (af-ZA) culture has changed from "nm" to "PM".	Note culture name changes.

Feature	Differences from 3.5 SP1	Recommended changes
LCID parameter	To be consistent with expected behavior in automation server settings, the CLR no longer passes the current culture for the <code>LCID</code> parameter to unmanaged COM-based applications. Instead, it passes 1033 (en-us) for the culture.	No modifications necessary except for native applications that require a specified culture.
Obsolete culture types	The <code>CultureTypes</code> and <code>CultureTypes</code> culture types are now obsolete. For backward compatibility, <code>CultureTypes</code> now returns neutral and specific cultures that were included with the previous .NET Framework, and <code>CultureTypes</code> now returns an empty list.	Use other values of the <code>CultureTypes</code> enumeration.
Retrieving culture	Beginning with Windows 7, .NET Framework 4 retrieves culture information from the operating system instead of storing the data itself. In addition, .NET Framework synchronizes with Windows for sorting and casing data.	None.
Unicode 5.1 standards	.NET Framework now supports all Unicode 5.1 characters -- an addition of approximately 1400 characters. The additional characters include new symbols, arrows, diacritics, punctuation, mathematical symbols, CJK strokes and ideographs, additional Malayalam and Telugu numeric characters, and various Myanmar, Latin, Arabic, Greek, Mongolian, and Cyrillic characters. The following new scripts are supported with Unicode 5.1: Sundanese, Lepcha, Ol Chiki, Vai, Saurashtra, Kayah Li, Rejang, Gurmukhi, Odia, Tamil, Telugu, and Malayalam characters and Cham.	None.

Exceptions

Namespaces: [System](#), [System.Runtime.ExceptionServices](#)

Assembly: mscorlib (in mscorlib.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Exceptions for corrupted process state	The CLR no longer delivers exceptions for corrupted process state to exception handlers in managed code.	These exceptions indicate that the state of a process has been corrupted. We do not recommend that you run your application in this state. For more information, see the HandleProcessCorruptedStateExceptionsAttribute and the entry Handling Corrupted State Exceptions in the MSDN magazine.
Execution engine exceptions	<code>ExecutionEngineException</code> is now obsolete, because a catchable exception will allow an unstable process to continue to run. This change improves predictability and reliability in the runtime.	Use an <code>InvalidOperationException</code> to signal the condition.

Reflection

Namespace: [System.Reflection](#)

Assembly: mscorlib (in mscorlib.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Assembly hash algorithms	The HashAlgorithm property now returns AssemblyHashAlgorithm , because the runtime does not know the hash algorithm of the referenced assembly when the assembly is not loaded. (This refers to using the property on a referenced assembly such as that returned by the GetReferencedAssemblies method.)	None.
Assembly loading	To prevent redundant loading of assemblies and to save virtual address space, the CLR now loads assemblies by using only the Win32 MapViewOfFile function. It no longer also calls the LoadLibrary function. This change affects diagnostic applications in the following ways: <ul style="list-style-type: none"> * A ProcessModuleCollection will no longer contain any modules from a class library (.dll file), as obtained from a call to Process.GetCurrentProcess().Modules. * Win32 applications that use the EnumProcessModules function will not see all managed modules listed. 	None.
Declaring type	The DeclaringType property now correctly returns null when the type does not have a declaring type.	None.
Delegates	A delegate now throws an ArgumentNullException instead of a NullReferenceException when a null value is passed to the delegate's constructor.	Ensure that any exception handling catches ArgumentNullException .
Global assembly cache location change	For .NET Framework 4 assemblies, the global assembly cache has been moved from the Windows directory (%WINDIR%) to the Microsoft.Net subdirectory (%WINDIR%\Microsoft.Net). Assemblies from earlier versions remain in the older directory. The unmanaged ASM_CACHE_FLAGS enumeration contains the new ASM_CACHE_ROOT_EX flag. This flag gets the cache location for .NET Framework 4 assemblies, which can be obtained by the GetCachePath function.	None, assuming that applications do not use explicit paths to assemblies, which is not a recommended practice.
Global assembly cache tool	The Gacutil.exe (Global Assembly Cache Tool) no longer supports the shell plugin viewer.	None.

Interoperability

Namespace: [System.Runtime.InteropServices](#)

Assembly: mscorelib (in mscorelib.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Buffer length (unmanaged API)	To save memory, the functionality for the pBufferLengthOffset parameter for the ICorProfilerInfo2::GetStringLayout method has been changed to match the pStringLengthOffset parameter. Both parameters now point to the offset location of the string's length. Buffer length has been removed from the representation of the string class.	Remove any dependency on the buffer length.

Feature	Differences from 3.5 SP1	Recommended changes
JIT debugging	To simplify registration for just-in-time (JIT) debugging, the .NET Framework debugger now uses only the AeDebug registry key, which controls the JIT debugging behavior for native code. This change results in the following: <ul style="list-style-type: none"> * You can no longer register two different debuggers for managed and native code. * You can no longer start the debugger automatically for a non-interactive process, but you can prompt the user for an interactive process. * You are no longer notified when the debugger fails to start, or when there is no registered debugger that should be started. * Auto-launch policies that depend on the interactivity of the application are no longer supported. 	Adjust debugging operations as required.
Platform invoke	To improve performance in interoperability with unmanaged code, incorrect calling conventions in a platform invoke now cause the application to fail. In previous versions, the marshalling layer resolved these errors up the stack.	Debugging your applications in Microsoft Visual Studio alerts you to these errors so you can correct them. If you have binaries that cannot be updated, you can include the <code><NetFx40_PInvokeStackResilience></code> element in your application's configuration file to enable calling errors to be resolved up the stack as in earlier versions. However, this may affect the performance of your application.
Removed interfaces (unmanaged API)	To avoid developer confusion, the following interfaces were removed, because they did not provide any useful run-time scenarios, and the CLR did not provide or accept any implementations: <ul style="list-style-type: none"> * <code>INativeImageNativeImageDependency</code> * <code>INativeImageInstallInfo</code> * <code>INativeImageEvaluate</code> * <code>INativeImageConverter</code> * <code>ICorModule</code> * <code>IMetaDataConverter</code> 	None.

Data

This section describes migration issues for using data sets and SQL clients, the Entity Framework, LINQ to SQL, and WCF Data Servers (formerly known as ADO.NET Data Services).

DataSet and SQL Client

The following table describes improvements to features that previously had limitations or other issues.

Namespaces: [System.Data](#), [System.Data.Objects.DataClasses](#), [System.Data.SqlClient](#)

Assemblies: System.Data (in System.Data.dll), System.Data.Entity (in System.Data.Entity.dll)

Feature	Differences from 3.5 SP1
---------	--------------------------

Feature	Differences from 3.5 SP1
POCO Scenarios	The IRelatedEnd interface has new methods to improve its usability in Plain Old CLR Object (POCO) scenarios. These new methods take an Object instead of an IEntityWithRelationships entity as a parameter.
Editing Rows	The IndexOf method, as implemented by the DataView class, now correctly returns the value of a row that is being edited, instead of returning -1.
Events	The PropertyChanged event is now raised when a row is in a modified state and the RejectChanges method is called. This change makes it easier to create UI controls that expose the contents of a DataSet object.
Exceptions	The Prepare method now throws an InvalidOperationException when a connection is not set or open instead of a NullReferenceException .
Mapping Views	Query view mapping errors are now caught at design time instead of throwing a NullReferenceException at run time. Mapping validation now catches the error in which two association sets in Conceptual Schema (CSDL) are mapped to the same column.
Transactions	If an application tries to execute a statement on a connection after a transaction has been completed (including aborted or rolled back), an InvalidOperationException is now thrown. Previous versions did not throw an exception and let you execute additional commands even if a transaction was aborted.

Entity Framework

The following table describes improvements to features that previously had limitations or other issues.

Namespaces: [System.Data](#), [System.Data.Objects](#), [System.Data.Objects.DataClasses](#)

Assemblies: System.Data.Entity (in System.Data.Entity.dll)

Feature	Differences from 3.5 SP1
Entity objects	There is now parity between the Detach method and the state of the entity object when the SaveChanges method is called. This improved consistency prevents unexpected exceptions from being thrown.
Entity SQL	Rules have been improved for identifier resolutions in Entity SQL. The Entity SQL parser has improved logic for resolving multipart identifiers.
Structural annotations	The Entity Framework now recognizes structural annotations.
Queries	The following improvements were made in queries: * A GroupBy query using a null key over an empty collection will not return any rows, regardless if there are any additional selects in the query. * Generated SQL in LINQ and Entity-SQL queries now treat string parameters as non-Unicode values by default.

LINQ to SQL

The following table describes improvements to features that previously had limitations or other issues.

Namespace: [System.Data.Linq](#)

Assembly: System.Data.Linq (in System.Data.Linq.dll)

Feature	Differences from 3.5 SP1
Events	A <code>EntitySet< TEntity ></code> collection now raises the <code>ListChanged</code> event for add and remove operations if the <code>EntitySet< TEntity ></code> is unloaded, in addition to raising the event when the collection is loaded.
Queries	<code>Skip(0)</code> is no longer ignored in LINQ to SQL queries. As a result, queries that have this method might behave differently. For example, in some cases, an <code>OrderBy</code> clause is required with <code>Skip(0)</code> , and the query will now throw a <code>NotSupportedException</code> exception if the <code>OrderBy</code> clause was not included.

WCF Data Services

The following table describes improvements to features that previously had limitations or other issues.

Namespaces: [System.Data.Services](#), [System.Data.Services.Client](#), [System.Data.Services.Common](#), [System.Data.Services.Providers](#)

Assemblies: System.Data.Services (in System.Data.Services.dll), System.Data.Services.Client (in System.Data.Services.Client.dll)

Feature	Differences from 3.5 SP1
Batched Binary Content	WCF Data Services now supports batched binary content in requests and responses.
Change interceptors	Change interceptors are now executed for a delete request. A change interceptor is a method that runs every time a request is received by the server to modify an entity in the entity set. It runs before the incoming request is executed. The change interceptor provides access to the entity that is being changed and the operation that is being performed on it.
Exceptions	The following conditions now throw more useful exceptions instead of a <code>NullReferenceException</code> : <ul style="list-style-type: none"> * A <code>TimeoutException</code> when a call to a data service times out. * A <code>DataServiceRequestException</code> when a bad request is made to a data service. In your applications, you should change exception handling to catch the new exceptions.
Headers	The following improvements were made to headers: <ul style="list-style-type: none"> * WCF Data Services now correctly rejects an <code>eTag</code> header that has an unspecified value. * WCF Data Services now returns an error and does not execute the request for a delete request to a link when an <code>if-</code> header is in the request. * WCF Data Services now returns an error to the client in the format (Atom, JSON) that the client specified in the <code>Accept</code> header.
JSON Reader	The JavaScript Object Notation (JSON) reader now correctly returns an error when it reads the single backslash ("\\") escape character, when it processes JSON payloads sent to a WCF Data Service.
Merges	The following improvements were made to the <code>MergeOption</code> enumeration: <ul style="list-style-type: none"> * The <code>MergeOption</code> merge option no longer modifies the entity on the client as a result of any subsequent response from a data service. * The <code>MergeOption</code> option is now consistent between dynamic SQL and stored procedure-based updates.
Requests	The <code>OnStartProcessingRequest</code> method is now called before a request to data services is processed. This enables the request to work correctly for <code>ServiceOperation</code> services.
Streams	WCF Data Services no longer closes the underlying stream for read and write operations.

Feature	Differences from 3.5 SP1
URIs	The escaping of URIs by WCF Data Services client has been corrected.

Windows Communication Foundation (WCF)

The following table describes improvements to features that previously had limitations or other issues.

Feature	Differences from 3.5 SP1
Configuration files	<p>To enable inheritance of behaviors through the configuration file hierarchy, WCF now supports merging across configuration files.</p> <p>The configuration inheritance model is now expanded to let users define behaviors that will be applied to all the services that are running on the computer.</p> <p>You may encounter behavioral changes if there are behaviors with the same name at different levels of the hierarchy.</p>
Service hosting	<p>You can no longer specify the <code><serviceHostingEnvironment></code> configuration element at the service level by adding the attribute <code>allowDefinition="MachineToApplication"</code> to the element definition.</p> <p>Specifying the <code><serviceHostingEnvironment></code> element at the service level is technically incorrect and causes inconsistent behavior.</p>

Windows Presentation Foundation (WPF)

Applications

Namespaces: [System.Windows](#), [System.Windows.Controls](#)

Assemblies: PresentationFramework (in PresentationFramework.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Exception handling	To enable errors to be detected earlier, WPF throws a TargetInvocationException and sets the InnerException property to critical exceptions, such as NullReferenceException , OutOfMemoryException , StackOverflowException , and SecurityException , instead of catching the original exception.	None.
Linked resources	To make linking easier, resource files (such as images) that are located in a location other than the project's folder structure use the resource file's full path instead of just its file name as the resource ID when the application is built. The application will be able to locate the files at run time.	None.

Feature	Differences from 3.5 SP1	Recommended changes
Partial-trust applications	<p>For security considerations, Windows-based applications that run in partial trust and contain a WebBrowser control or a Frame control that contains HTML will throw a SecurityException when the control is created.</p> <p>Browser applications will throw an exception and display a message if all of the following conditions are met:</p> <ul style="list-style-type: none"> * The application is running in Firefox. * The application is running in partial trust in the Internet zone from non-trusted sites. * The application contains a WebBrowser control or a Frame control that contains HTML. <p>Applications that run from trusted sites or from the intranet zone will not be affected.</p>	<p>In your browser applications, you can ease this change by doing one of the following:</p> <ul style="list-style-type: none"> * Run the browser application in full trust. * Have customers add the application's site to the trusted sites zone.
Resource dictionaries	<p>To improve theme-level resource dictionaries and prevent them from changing, freezable resources that are defined in a resource dictionary and merged into a theme-level dictionary are now always marked as frozen and are immutable. This is the expected behavior for freezable resources.</p>	<p>Applications that modify a resource that is defined in a theme-level merged dictionary should clone the resource and modify the cloned copy. Alternatively, the resource can be marked <code>x:Shared="false"</code> so that the ResourceDictionary creates a new copy every time the resource is queried.</p>
Windows 7	<p>To make WPF applications work better on Windows 7, the following improvements were made to correct the behavior of a window:</p> <ul style="list-style-type: none"> * Dock and gesture states now work as expected based on user interactions. * The taskbar commands Cascade windows, Show windows stacked, and Show windows side-by-side now have the correct behavior and update the appropriate properties. * The <code>Top</code>, <code>Left</code>, <code>Width</code>, and <code>Height</code> properties for a maximized or minimized window now contain the correct restore location of the window instead of other values, depending on the monitor. 	None.
Windows style and transparency	<p>An InvalidOperationException is thrown if you try to set WindowStyle to a value other than WindowState when AllowsTransparency is <code>true</code> and WindowState is WindowState.</p>	<p>If you must change the WindowStyle when AllowsTransparency is <code>true</code>, you can call the Win32 <code>SetWindowLongPtr</code> function.</p>
XPS Viewer	<p>WPF does not include the Microsoft XML Paper Specification Essentials Pack (XPSEP). XPSEP is included with Windows 7 and Windows Vista.</p> <p>On a computer that is running Windows XP without .NET Framework 3.5 SP1 installed, printing by using a WPF API other than those in PrintDialog will rely on the WINSPOOL. Some printer capabilities will not be reported and some printer settings will not be applied during printing.</p>	<p>If needed, install the Microsoft XML Paper Specification Essentials Pack.</p>

Controls

Namespaces: [System.Windows](#), [System.Windows.Controls](#), [System.Windows.Data](#), [System.Windows.Input](#)

Assemblies: PresentationFramework (in PresentationFramework.dll), PresentationCore (in PresentationCore.dll), WindowsBase (in WindowsBase.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Dialog boxes	To improve reliability, the ShowDialog method is called on the same thread that created the FileDialog control.	Be sure to create a FileDialog control and call the ShowDialog method on the same thread.
Floating windows	To fix focus restoration logic that incorrectly keeps reactivating a floating window (making it appear like a modal dialog box), focus restoration is now prevented if the candidate is not a child of the window.	None.
Items in collections	When an item is moved or added to an underlying collection, it appears in the CollectionView in the same relative location if the CollectionView is not sorted. This provides consistency between the item's position in the collection and in the associated CollectionView .	Use the ContainerFromItem or IndexOf method to find the location of an item in a CollectionView instead of relying on a fixed location of an item.
Layouts	To eliminate unnecessary re-layouts, changing the ShowsNavigationUI no longer invalidates layout or causes another layout pass.	If you expect that changing ShowsNavigationUI will cause another layout pass, call InvalidateVisual after you set the property.
Menus	To enable ClearType text in menu pop-ups, modifications were made to the ControlTemplate class and to the MenuItem control and other controls.	Applications should not rely on the visual structure of control templates. Only named parts of a ControlTemplate are part of the public contract. If an application must find a certain object in a ControlTemplate , search the visual tree for a specific type instead of relying on a fixed location of an object in the tree.
Navigating	If a Frame directly navigates to a location, the IsNavigationInitiator property is <code>true</code> after the initial navigation. This change prevents additional events from being raised during startup scenarios.	None.
Pop-ups	The CustomPopupPlacementCallback delegate can now be called multiple times during a layout pass instead of only once.	If your CustomPopupPlacementCallback delegate calculates the position of a Popup based on its previous position, recalculate the position only if the values of the <code>popupSize</code> , <code>targetSize</code> , or <code>offset</code> parameters change.
Property values	The SetCurrentValue method now lets you set a property to an effective value, although it continues to respect any binding, style, or trigger that affects the property.	Control authors should use SetCurrentValue whenever the property value changes as a side-effect of some other action, including user manipulation.
Text boxes	For security considerations, the Copy and Cut methods silently fail when they are called in partial trust. In addition, programmatic execution of the Copy or Cut property on a control that inherits from TextBoxBase will be blocked in partial trust. However, user-initiated copy and cut commands, such as clicking a button whose Command property is bound to one of these commands, will work. Standard copy and cut through keyboard shortcuts and the context menu will still work as before in partial trust.	Bind the Copy or Cut command to a user-initiated action, such as clicking a button.

Graphics

Namespaces: [System.Windows](#), [System.Windows.Controls](#), [System.Windows.Data](#), [System.Windows.Input](#), [System.Windows.Media.Effects](#)

Assemblies: PresentationFramework (in PresentationFramework.dll), PresentationCore (in PresentationCore.dll), WindowsBase (in WindowsBase.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Bitmap effects	<p>To improve performance, the BitmapEffect class and the classes that inherit from the BitmapEffect class, although still present, are disabled. The effect will render by using the hardware-accelerated rendering pipeline if the following conditions are true:</p> <ul style="list-style-type: none">* The application uses a DropShadowBitmapEffect or a BlurBitmapEffect that has a radius property set less than 100 DIU.* The video card on the computer that runs the application supports pixel shader 2.0. <p>If these conditions are not met, a BitmapEffect object will have no effect.</p> <p>Also, Visual Studio produces a compiler warning when it encounters the BitmapEffect object or a subclass.</p> <p>The PushEffect method is marked obsolete.</p>	<p>Discontinue using the legacy BitmapEffect and derived classes and instead use the new classes derived from Effect: BlurEffect, DropShadowEffect, and ShaderEffect.</p> <p>You can also create your own effects by inheriting from the ShaderEffect class.</p>
Bitmap frames	<p>The cloned BitmapFrame objects now receive the DownloadProgress, DownloadCompleted, and DownloadFailed events. This enables images that are downloaded from the Web and applied to the Image control through a Style to work correctly.</p> <p>You will see a change in behavior only if all of the following statements are true:</p> <ul style="list-style-type: none">* You subscribe to the DownloadProgress, DownloadCompleted, or DownloadFailed event.* The source of the BitmapFrame is from the Web.* The BitmapFrame is cloned while the download is still in progress.	<p>Check the sender in the event handler and take action only if the sender is the original BitmapFrame.</p>
Decoding images	<p>To prevent an IOException from not being handled when images may not decode, the BitmapSource class will raise the DecodeFailed event when it does not decode an image.</p>	<p>Remove any exception handling for IOException, and use the DecodeFailed event to check for decode failure.</p>

Input

Namespaces: [System.Windows](#), [System.Windows.Controls](#), [System.Windows.Data](#), [System.Windows.Input](#)

Assemblies: PresentationFramework (in PresentationFramework.dll), PresentationCore (in PresentationCore.dll), WindowsBase (in WindowsBase.dll)

Feature	Differences from 3.5 SP1	Recommended changes
---------	--------------------------	---------------------

Feature	Differences from 3.5 SP1	Recommended changes
Binding command instances	<p>To provide a mechanism to bind View-Model-based command instances to View-based input gestures, the InputBinding class now inherits from Freezable instead of DependencyObject. The following properties are now dependency properties:</p> <ul style="list-style-type: none"> * Command * CommandParameter * CommandTarget <p>This change results in the following:</p> <ul style="list-style-type: none"> * An InputBinding object is now frozen when it is registered instead of remaining mutable. * You cannot access instance-level InputBinding objects from multiple threads, due to the restrictions of the DependencyObject class. * You cannot mutate class-level input bindings after their registration, due to the restrictions of the Freezable class. * You cannot specify input bindings on command instances that are created in a View-Model. 	Create separate instances of an InputBinding class on separate threads if bindings are to be mutable, or freeze them otherwise. Do not mutate a class-level static InputBinding after it has been registered.
Browser applications	WPF Browser applications (.XBAP) now process key events just like stand-alone WPF applications so that objects receive routed key events in the correct order.	None.
Dead key combinations	WPF obfuscates dead keys, which produce no visible character, but instead indicates that the key is to be combined with the next letter key to produce one character. The key input events, such as the KeyDownEvent event, report when a key is a dead key by setting the Key property to the Key value. This is usually expected behavior because applications usually do not intend to respond to keyboard input that creates a combined character.	Applications that expect to read keys that were part of combined characters can get the now obfuscated key by using the DeadCharProcessedKey property.
Focus manager	When the FocusManager.GetFocusedElement(DependencyObject) method is passed an element that has the IsFocusScope attached property set to <code>true</code> , the method returns an element that is the last keyboard-focused element within that focus scope if and only if the returned element belongs to the same PresentationSource object as the element that is passed to the method.	None.

UI Automation

Namespace: [System.Windows](#), [System.Windows.Automation.Peers](#), [System.Windows.Automation.Provider](#), [System.Windows.Controls](#), [System.Windows.Data](#), [System.Windows.Input](#)

Assemblies: PresentationFramework (in PresentationFramework.dll), PresentationCore (in PresentationCore.dll), UIAutomationProvider (in UIAutomationProvider.dll), WindowsBase (in WindowsBase.dll)

Feature	Differences from 3.5 SP1	Recommended changes
Class hierarchy of views	The TreeViewAutomationPeer and TreeViewItemAutomationPeer classes inherit from ItemsControlAutomationPeer instead of FrameworkElementAutomationPeer .	If you inherit from the TreeViewItemAutomationPeer classes and override the GetChildrenCore method, consider returning an object that inherits from the new TreeViewDataItemAutomationPeer class.

Feature	Differences from 3.5 SP1	Recommended changes
Containers off screen	To fix an incorrect return value, the IsOffscreenCore method now correctly returns <code>false</code> for item containers that are scrolled out of view. Also, the value of the method is not affected by occlusion by other windows, or by whether the element is visible on a specific monitor.	None.
Menus and child objects	To enable UI automation of menus that contain children other than MenuItem objects, the GetChildrenCore method now returns the AutomationPeer object of a child UIElement object, instead of a MenuItemAutomationPeer object.	None.
New interfaces and assembly	To enable new features for UI automation, the following interfaces were added: * IItemContainerProvider * ISynchronizedInputProvider * IVirtualizedItemProvider	Any project that builds WPF automation peers must add an explicit reference to UIAutomationProvider.dll .
Thumbs	The GetClassNameCore method returns a value instead of null. Therefore, controls such as GridSplitter that inherit from the Thumb class will report a name to UI Automation.	None.
Virtualized elements	To improve performance, the GetChildrenCore method returns only the child objects that are actually in the visual tree, instead of all child objects, regardless of whether they are virtualized.	Use ItemContainerPattern to iterate over all items of an ItemsControlAutomationPeer .

XAML

Namespaces: [System.Windows](#), [System.Windows.Controls](#), [System.Windows.Data](#), [System.Windows.Input](#), [System.Windows.Markup](#)

Assemblies: PresentationFramework (in [PresentationFramework.dll](#)), PresentationCore (in [PresentationCore.dll](#)), WindowsBase (in [WindowsBase.dll](#))

Feature	Differences from 3.5 SP1	Recommended changes
Markup extension	WPF now correctly always uses the value from the ProvideValue method instead of returning the MarkupExtension object in certain cases when a markup extension is used to set a property or to create an item in a collection. A markup extension might return itself in some cases.	If your application accesses a resource that returned a MarkupExtension object in earlier versions, reference the object that is returned from ProvideValue , instead of the MarkupExtension object.
Parsing attributes	Attributes in XAML can now have only one period. For example, the following are valid: <pre><Button Background="Red"/> (no periods)</pre> <pre><Button Button.Background = "Red"/> (one period)</pre> The following is no longer valid: <pre><Button Control.Button.Background = "Red"/> (more than one period)</pre>	Correct XAML attributes that have more than one period.

XML

The rows in this table describe improvements to features that previously had limitations or other issues.

Schema and transforms

Namespaces: [System.Xml.Linq](#); [System.Xml.Schema](#), [System.Xml.XPath](#)

Assemblies: System.Xml (in System.Xml.dll), System.Xml.Linq (in System.Xml.Linq.dll)

Feature	Differences from 3.5 SP1
Chameleon schemas	To prevent data corruption, chameleon schemas are now cloned correctly when they are included with multiple schemas. Chameleon schemas are schemas that do not have a target namespace, and when they are included in another XSD, they take on the target namespace of the importing schema. They are often used to include common types into a schema.
ID functions	The XSLT id function now returns the correct value instead of null when an XmlReader object is passed to an XSLT. If the user created an XmlReader object from a LINQ to XML class by using the CreateReader method, and this XmlReader object was passed to an XSLT, any instances of the <code>id</code> function in the XSLT previously returned null. This is not an allowed return value for the <code>id</code> function.
Namespace attribute	To prevent data corruption, an XPathNavigator object now returns the local name of the <code>x:xmlns</code> attribute correctly.
Namespace declarations	An XmlReader object on a subtree no longer creates duplicate namespace declarations within one XML element.
Schema validation	To prevent erroneous schema validation, the XmlSchemaSet class allows for XSD schemas to be compiled correctly and consistently. These schemas can include other schemas; for example, <code>A.xsd</code> can include <code>B.xsd</code> , which can include <code>C.xsd</code> . Compiling any one of these causes this graph of dependencies to be traversed.
Script functions	The function-available function no longer incorrectly returns <code>false</code> when the function is actually available.
URIs	The Load method now returns the correct BaseURI in LINQ queries.

Validation

Namespaces: [System.Xml.Linq](#); [System.Xml.Schema](#), [System.Xml.XPath](#)

Assemblies: System.Xml (in System.Xml.dll), System.Xml.Linq (in System.Xml.Linq.dll)

Feature	Differences from 3.5 SP1
Namespace resolvers	The ReadContentAs method no longer ignores the IXmlNamespaceResolver resolver passed to it. In previous versions, the specified namespace resolver was ignored, and the XmlReader was used instead.
White space	To prevent data loss when you are creating a reader, the Create method no longer discards significant white space. XML validation recognizes mixed-content mode, where text can be intermixed with XML markup. In mixed mode, all white space is significant and should be reported.

Writing

Namespaces: [System.Xml.Linq](#); [System.Xml.Schema](#), [System.Xml.XPath](#)

Assemblies: System.Xml (in System.Xml.dll), System.Xml.Linq (in System.Xml.Linq.dll)

Feature	Differences from 3.5 SP1
Entity references	To prevent data corruption, entity references are no longer entitized twice in XML attributes. If the user tried to write an entity into an <code>xmlns</code> attribute or into an <code>xml:lang</code> or <code>xml:space</code> attribute by using the WriteEntityRef method, the entity was entitized twice in the output, therefore corrupting the data.
New line handling	To prevent data corruption, XmlWriter objects respect the NewLineHandling option.

See also

- [New Types and Members in .NET Framework 4](#)
- [Migration Guide to .NET Framework 4](#)
- [What's New in .NET Framework 4](#)
- [Version Compatibility in .NET Framework](#)
- [Migrating Office Solutions to .NET Framework 4](#)
- [What's Obsolete in the .NET Framework Class Library](#)

Build apps against Microsoft.NETFramework.ReferenceAssemblies

Article • 04/30/2024

When you target a particular version of .NET Framework, by default your application is built by using the reference assemblies that are included with that version's developer pack. In scenarios where the matching developer pack can't be installed on the computer, you can build against reference assemblies distributed via a NuGet package instead.

Update project files

Each project that should build against the reference assemblies NuGet package needs to include a reference to *Microsoft.NETFramework.ReferenceAssemblies*.

Projects that use a *packages.config* file should include the following in *packages.config*.

XML

```
<packages>
  <package id="Microsoft.NETFramework.ReferenceAssemblies" version="1.0.3"
developmentDependency="true" />
</packages>
```

Projects that use the *<PackageReference>* MSBuild property should include the following property in the project file.

XML

```
<ItemGroup>
  <PackageReference Include="Microsoft.NETFramework.ReferenceAssemblies"
Version="1.0.3" PrivateAssets="All" />
</ItemGroup>
```

SDK-style projects include this reference by default. For typical .NET Framework projects that were created with Visual Studio, you can add the reference by using the NuGet Package Manager UI in Visual Studio. The package contains reference assemblies for many versions of .NET Framework. The version that's actually used is determined by the

`TargetFrameworkVersion` or `TargetFramework` property, as already defined in the project file.

Restore the project

Projects that contain a package reference must be restored before they can be built.

After adding the `Microsoft.NETFramework.ReferenceAssemblies` NuGet package to your project, you must explicitly run the restore action in one of the following ways:

- If your project is a non SDK-style project and uses the `packages.config` file to reference NuGet packages:
 1. Install the [NuGet CLI tool](#), and make sure `nuget.exe` is in the `PATH` environment variable.
 2. Open a command prompt.
 3. Navigate to the directory that contains your project file.
 4. Run `nuget.exe restore`.
- If your project is a non SDK-style project and uses `<PackageReference>` settings in the project file to reference NuGet packages:
 1. Open **Developer Command Prompt for VS 2022**. The name of this app might be different based on which version of Visual Studio you've installed.
 2. Navigate to the directory that contains your project file.
 3. Run `msbuild /t:restore`.
- If your project is an SDK-style project, you don't need to do anything. The NuGet restore action runs automatically when the project is built.

Important

Using reference assemblies makes it possible to build projects that target unsupported versions of .NET Framework from the command line. However, you still can't load these projects in newer versions of Visual Studio. To continue building these apps in Visual Studio, the only workaround is to use [an older version of Visual Studio](#).



Collaborate with us on



.NET feedback

GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

.NET Framework development guide

Article • 08/27/2022

This section explains how to create, configure, debug, secure, and deploy your .NET Framework apps. The section also provides information about technology areas such as dynamic programming, interoperability, extensibility, memory management, and threading.

In this section

[Data and modeling](#) Provides information about how to access data using ADO.NET, Language Integrated Query (LINQ), WCF Data Services, and XML.

[Client applications](#) Explains how to create Windows-based apps by using Windows Presentation Foundation (WPF) or Windows Forms.

[Web applications with ASP.NET](#) Provides links to information about using ASP.NET to build enterprise-class web apps with a minimum of coding.

[Service-Oriented Applications with WCF](#) Describes how to use Windows Communication Foundation (WCF) to build service-oriented apps that are secure and reliable.

[Building workflows with Windows Workflow Foundation](#) Provides information about the programming model, samples, and tools for using Windows Workflow Foundation (WF).

[Windows service applications](#) Explains how you can use Visual Studio and the .NET Framework to create an app that is installed as a service, and start, stop, and otherwise control its behavior.

[Parallel Processing in .NET](#) Provides information about parallel programming.

[Concurrency in .NET](#) Provides information about managed threading.

[Async programming patterns in .NET](#) Provides information about asynchronous programming design patterns.

[Network programming in the .NET](#) Describes the layered, extensible, and managed implementation of Internet services that you can quickly and easily integrate into your apps.

[Configuring .NET Framework apps](#) Explains how you can use configuration files to change settings without having to recompile your .NET Framework apps.

[Security](#) Provides information about the classes and services in the .NET Framework that facilitate secure app development.

[Debugging, Tracing, and Profiling](#) Explains how to test, optimize, and profile .NET Framework apps and the app environment. This section includes information for administrators as well as developers.

[Developing for multiple platforms](#) Provides information about how you can use the .NET Framework to build assemblies that can be shared across multiple platforms and multiple devices such as phones, desktop, and web.

[Deployment](#) Explains how to package and distribute your .NET Framework app, and includes deployment guides for both developers and administrators.

[Performance](#) Provides information about caching, lazy initialization, reliability, and ETW events.

Reference

[.NET Framework class library](#) Supplies syntax, code examples, and usage information for each class that is contained in the .NET Framework namespaces.

Related sections

[Get started](#) Provides a comprehensive overview of the .NET Framework and links to additional resources.

[What's new](#) Describes key new features and changes in the latest version of the .NET Framework. Includes lists of new and obsolete types and members, and provides a guide for migrating your apps from the previous version of the .NET Framework.

[Tools](#) Describes the tools that help you develop, configure, and deploy apps by using .NET Framework technologies.

[.NET samples and tutorials](#) Provides links to samples and tutorials that help you learn about .NET.

Programming with Application Domains and Assemblies

Article • 06/04/2024

ⓘ Note

This article is specific to .NET Framework. It doesn't apply to newer implementations of .NET, including .NET 6 and later versions.

Hosts such as ASP.NET and the Windows shell load the common language runtime into a process, create an [application domain](#) in that process, and then load and execute user code in that application domain when running a .NET Framework application. In most cases, you do not have to worry about creating application domains and loading assemblies into them because the runtime host performs those tasks.

However, if you're creating an application that will host the common language runtime, creating tools or code you want to unload programmatically, or creating pluggable components that can be unloaded and reloaded on the fly, you will be creating your own application domains. Even if you are not creating a runtime host, this section provides important information on how to work with application domains and assemblies loaded in these application domains.

In This Section

[Using Application Domains](#)

Provides examples of creating, configuring, and using application domains.

[Programming with Assemblies](#)

Describes how to create, sign, and set attributes on assemblies.

Related Sections

[Emitting Dynamic Methods and Assemblies](#)

Describes how to create dynamic assemblies.

[Assemblies in .NET](#)

Provides a conceptual overview of assemblies.

[Application Domains](#)

Provides a conceptual overview of application domains.

[Reflection Overview](#)

Describes how to use the `Reflection` class to obtain information about an assembly.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Microsoft UI Automation

Article • 09/15/2021

ⓘ Note

This documentation is intended for .NET developers who want to use the managed UI Automation classes defined in the [System.Windows.Automation](#) namespace. For the latest information about UI Automation, see [Windows Automation API: UI Automation](#).

Microsoft UI Automation is an accessibility framework for Microsoft Windows. It addresses the needs of assistive technology products and automated test frameworks by providing programmatic access to information about the user interface (UI). In addition, UI Automation enables control and application developers to make their products accessible.

This documentation describes the UI Automation API for managed code. For information on programming for UI Automation in C++, see [UI Automation for Win32 Applications](#).

In this section

- [Accessibility Best Practices](#)
- [UI Automation Fundamentals](#)
- [UI Automation Providers for Managed Code](#)
- [UI Automation Clients for Managed Code](#)
- [UI Automation Control Patterns](#)
- [UI Automation Text Pattern](#)
- [UI Automation Control Types](#)

Related sections

- [Accessibility Samples ↗](#)

Data and modeling in .NET

Article • 11/10/2021

This section provides information on how to access data in .NET Framework.

In this section

[ADO.NET](#)

Describes the ADO.NET architecture and how to use the ADO.NET classes to manage application data and interact with data sources, including Microsoft SQL Server, OLE DB data sources, and XML.

[Transaction Processing](#)

Discusses the .NET support for transactions.

Related sections

[Language Integrated Query \(LINQ\)](#)

Provides links to relevant documentation for Language Integrated Query (LINQ) using C#.

[Language-Integrated Query \(LINQ\) \(Visual Basic\)](#)

Provides links to relevant documentation for Language Integrated Query (LINQ) using Visual Basic.

[XML Documents and Data](#)

Provides an overview to a comprehensive and integrated set of classes that work with XML documents and data in .NET Framework.

[XML Standards Reference](#)

Provides reference information on XML standards that Microsoft supports.

Develop client applications with .NET Framework

Article • 09/15/2021

There are several ways to develop Windows-based applications with .NET Framework. You can use any of these tools and frameworks:

- [Universal Windows Platform \(UWP\)](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Forms](#)

This section contains articles that describe how to create Windows-based applications by using Windows Presentation Foundation or Windows Forms. However, you can also create web applications using .NET Framework and client applications for computers or devices that you make available through Microsoft Store (UWP apps).

Related sections

[Universal Windows Platform](#)

Describes how to create UWP applications that you can make available to users through Microsoft Store.

[.NET API for UWP apps](#)

Reference for .NET types that support UWP apps.

[Develop for Multiple Platforms](#)

Describes the different methods you can use .NET Framework to target multiple client app types.

[Get Started with ASP.NET Web Sites ↗](#)

Describes the ways you can develop web apps using ASP.NET.

[.NET API for Windows Phone Silverlight](#)

Lists .NET Framework APIs you can use for building apps with Windows Phone Silverlight.

See also

- [.NET Standard](#)
- [Overview](#)

- Development Guide
- Windows Service Applications

Windows Presentation Foundation

Article • 09/04/2020

Windows Presentation Foundation (WPF) provides developers with a unified programming model for building line-of-business desktop applications on Windows.

- [Introduction to WPF](#)
- [Getting Started](#)
- [Application Development](#)
- [Advanced](#)
- [Controls](#)
- [Data](#)
- [Graphics and Multimedia](#)
- [Security](#)
- [WPF Samples](#)
- [Class Library](#)

.NET Desktop Guide for Windows Forms

Learn about Windows Forms (WinForms), a graphical user interface for Windows and .NET Framework.

WinForms for .NET Framework

OVERVIEW

[Windows Forms overview](#)

GET STARTED

[Get started with Windows Forms Designer](#)

[Create a WinForms app from the command-line](#)

Controls

OVERVIEW

[Developing your own controls](#)

[Developing controls](#)

HOW-TO GUIDE

[Add Controls to Windows Forms](#)

[Position controls on Windows Forms](#)

REFERENCE

[Windows Forms Controls by Function](#)

Events

OVERVIEW

[Events in Windows Forms](#)

Event handlers in Windows Forms

CONCEPT

Order in which events are raised

Input

OVERVIEW

[About keyboard input](#)

[About mouse input](#)

CONCEPT

[Keyboard events](#)

[Mouse events](#)

[Mouse pointers](#)

HOW-TO GUIDE

[Modify keyboard input](#)

[Detect modifier keyboard keys](#)

[Distinguish between single/double clicks](#)

Develop Service-Oriented Applications with WCF

Article • 09/15/2021

This section of the documentation provides information about Windows Communication Foundation (WCF), which is a unified programming model for building service-oriented applications. It enables developers to build secure, reliable, transacted solutions that integrate across platforms and interoperate with existing investments.

In this section

[What's New in Windows Communication Foundation 4.5](#)

Discusses features new to Windows Communication Foundation.

[WCF Simplification Features](#)

Discusses new features that make writing WCF applications simpler.

[Guide to the Documentation](#)

A description of the WCF documentation

[Conceptual Overview](#)

Summarizes information about the Windows Communication Foundation (WCF) messaging system and the classes that support its use.

[Getting Started Tutorial](#)

A step by step tutorial to create a WCF service and client

[Basic WCF Programming](#)

Describes the fundamentals for creating Windows Communication Foundation applications.

[WCF Feature Details](#)

Show topics that let you choose which WCF feature or features you need to employ.

[Extending WCF](#)

Describes how to modify and extend WCF runtime components

[Guidelines and Best Practices](#)

Provides guidelines for creating Windows Communication Foundation (WCF) applications.

[Administration and Diagnostics](#)

Describes the diagnostic features of WCF

[Operating System Resources Required by WCF](#)

Describes operating system resources required by WCF

[Troubleshooting Setup Issues](#)

Provides guidance for fixing WCF setup issues

[Migrating from .NET Remoting to WCF](#)

Compares .NET Remoting to WCF and provides migration guidance for common scenarios.

[Using the WCF Development Tools](#)

Describes the Visual Studio Windows Communication Foundation development tools that can assist you in developing your WCFservice.

[Windows Communication Foundation Tools](#)

Describes WCF tools designed to make it easier to create, deploy, and manage WCF applications

[Windows Communication Foundation Samples](#)

Samples that provide instruction on various aspects of Windows Communication Foundation

[Windows Communication Foundation Glossary](#)

Shows a list of terms specific to WCF

[General Reference](#)

The section describes the elements that are used to configure Windows Communication Foundation clients and services.

[Privacy Information](#)

Information regarding WCF and Privacy

Windows Workflow Foundation

Article • 09/15/2021

This section describes the programming model, samples, and tools of Windows Workflow Foundation (WF).

In This Section

[Guide to the Windows Workflow Documentation](#)

A set of suggested topics to read, depending upon your familiarity (novice to well-acquainted), and requirements.

[What's New in Windows Workflow Foundation](#)

Discusses the changes in several development paradigms from previous versions.

[What's New in Windows Workflow Foundation in .NET Framework 4.5](#)

Describes the new features in Windows Workflow Foundation in .NET Framework 4.6.1.

[Windows Workflow Foundation Feature Specifics](#)

Describes the new features in Windows Workflow Foundation in .NET Framework 4.

[Windows Workflow Conceptual Overview](#)

A set of topics that discusses the larger concepts behind Windows Workflow Foundation.

[Getting Started Tutorial](#)

A set of walkthrough topics that introduce you to programming Windows Workflow Foundation applications.

[Windows Workflow Foundation Programming](#)

A set of primer topics that you should understand to become a proficient WF programmer.

[Extending Windows Workflow Foundation](#)

A set of topics that discusses how to extend or customize Windows Workflow Foundation to suit your needs.

[Windows Workflow Foundation Glossary for .NET Framework 4.5](#)

Defines a list of terms that are specific to WF.

[Windows Workflow Samples](#)

Contains sample applications that demonstrate WF features and scenarios.

Develop Windows service apps

Article • 09/15/2021

⚠ Warning

This documentation isn't for the latest version of Windows Service. For the latest content on Windows Services using **BackgroundService** and the Worker Service template, see:

- [Worker Services in .NET](#)
- [Create a Windows Service using BackgroundService](#)

Using Visual Studio or the .NET Framework SDK, you can easily create services by creating an application that is installed as a service. This type of application is called a Windows service. With framework features, you can create services, install them, and start, stop, and otherwise control their behavior.

ⓘ Note

In Visual Studio you can create a service in managed code in Visual C# or Visual Basic, which can interoperate with existing C++ code if required. Or, you can create a Windows service in native C++ by using the [ATL Project Wizard](#).

In this section

[Introduction to Windows Service Applications](#)

Provides an overview of Windows service applications, the lifetime of a service, and how service applications differ from other common project types.

[Walkthrough: Creating a Windows Service Application in the Component Designer](#)

Provides an example of creating a service in Visual Basic and Visual C#.

[Service Application Programming Architecture](#)

Explains the language elements used in service programming.

[How to: Create Windows Services](#)

Describes the process of creating and configuring Windows services using the Windows service project template.

Related sections

[ServiceBase](#) - Describes the major features of the `ServiceBase` class, which is used to create services.

[ServiceProcessInstaller](#) - Describes the features of the `ServiceProcessInstaller` class, which is used along with the `ServiceInstaller` class to install and uninstall your services.

[ServiceInstaller](#) - Describes the features of the `ServiceInstaller` class, which is used along with the `ServiceProcessInstaller` class to install and uninstall your service.

[Create Projects from Templates](#) - Describes the projects types used in this chapter and how to choose between them.

64-bit applications

Article • 06/04/2024

When you compile an application, you can specify that it should run on a Windows 64-bit operating system either as a native application or under WOW64 (Windows 32-bit on Windows 64-bit). WOW64 is a compatibility environment that enables a 32-bit application to run on a 64-bit system. WOW64 is included in all 64-bit versions of the Windows operating system.

Running 32-bit vs. 64-bit applications on Windows

32-bit applications that are built on .NET Framework 4 or later versions run under WOW64 on 64-bit systems.

Visual Studio installs the 32-bit version of the CLR on an x86 computer, and both the 32-bit version and the appropriate 64-bit version of the CLR on a 64-bit Windows computer. (Because Visual Studio is a 32-bit application, when it is installed on a 64-bit system, it runs under WOW64.)

ⓘ Note

Because of the design of x86 emulation and the WOW64 subsystem for the Itanium processor family, applications are restricted to execution on one processor. These factors reduce the performance and scalability of 32-bit .NET Framework applications that run on Itanium-based systems. We recommend that you use the .NET Framework 4, which includes native 64-bit support for Itanium-based systems, for increased performance and scalability.

By default, when you run a 64-bit managed application on a 64-bit Windows operating system, you can create an object of no more than 2 gigabytes (GB). However, in .NET Framework 4.5, you can increase this limit. For more information, see the [`<gcAllowVeryLargeObjects>` element](#).

Many assemblies run identically on both the 32-bit CLR and the 64-bit CLR. However, some programs may behave differently, depending on the CLR, if they contain one or more of the following:

- Structures that contain members that change size depending on the platform (for example, any pointer type).

- Pointer arithmetic that includes constant sizes.
- Incorrect platform invoke or COM declarations that use `Int32` for handles instead of `IntPtr`.
- Code that casts `IntPtr` to `Int32`.

For more information about how to port a 32-bit application to run on the 64-bit CLR, see [Migrating 32-bit Managed Code to 64-bit](#).

General 64-Bit Programming Information

For general information about 64-bit programming, see the following documents:

- In the Windows SDK documentation, see [Programming Guide for 64-bit Windows](#).
- For information about Visual Studio support for creating 64-bit applications, see [Visual Studio IDE 64-Bit Support](#).

Compiler Support for Creating 64-Bit Applications

By default, when you use the .NET Framework to build an application on either a 32-bit or a 64-bit computer, the application will run on a 64-bit computer as a native application (that is, not under WOW64). The following table lists documents that explain how to use Visual Studio compilers to create 64-bit applications that will run as native, under WOW64, or both.

[+] Expand table

Compiler	Compiler option
Visual Basic	-platform (Visual Basic)
Visual C#	-platform (C# Compiler Options)
Visual C++	You can create platform-agnostic, common intermediate language (CIL) applications by using <code>/clr:safe</code> . For more information, see -clr (Common Language Runtime Compilation) . Visual C++ includes a separate compiler for each 64-bit operating system. For more information about how to use Visual C++ to create native applications that run on a 64-bit Windows operating system, see 64-bit Programming .

Determining the Status of an .exe File or .dll File

To determine whether an .exe file or .dll file is meant to run only on a specific platform or under WOW64, use [CorFlags.exe \(CorFlags Conversion Tool\)](#) with no options. You can also use CorFlags.exe to change the platform status of an .exe file or .dll file. The CLR header of a Visual Studio assembly has the major runtime version number set to 2 and the minor runtime version number set to 5. Applications that have the minor runtime version set to 0 are treated as legacy applications and are always executed under WOW64.

To programmatically query an .exe or .dll to see whether it is meant to run only on a specific platform or under WOW64, use the [Module.GetPEKind](#) method.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Developing Web apps with ASP.NET

Article • 04/07/2022

ASP.NET is a .NET Framework technology for creating web apps. For more information on ASP.NET, see:

- [ASP.NET documentation](#)
- [ASP.NET MVC ↗](#)
- [ASP.NET Web Pages ↗](#)
- [ASP.NET Web API ↗](#)
- [Create an ASP.NET Framework web app in Azure](#)

Developing Web apps with ASP.NET Core

ASP.NET Core is a redesign of ASP.NET 4.x. Some of the benefits ASP.NET Core provides over ASP.NET:

- Cross platform.
- Leaner and more modular.
- A unified story for building web UI and web APIs.

See [Why use ASP.NET Core?](#) for an expanded list of benefits.

For more information on [ASP.NET Core](#), see:

- [Get started with Razor Pages](#)
- [Create a Web API](#)
- [Create an ASP.NET Core web app in Azure](#)

See also

- [Development Guide](#)

Configuring Internet Applications

Article • 03/30/2023

The [<system.Net> Element \(Network Settings\)](#) configuration element contains network configuration information for applications. Using the [<system.Net> Element \(Network Settings\)](#) element, you can set proxy servers, set connection management parameters, and include custom authentication and request modules in your application.

The [<defaultProxy> Element \(Network Settings\)](#) element defines the proxy server returned by the `GlobalProxySelection` class. Any `HttpWebRequest` that does not have its own `Proxy` property set to a specific value uses the default proxy. In addition to setting the proxy address, you can create a list of server addresses that will not use the proxy, and you can indicate that the proxy should not be used for local addresses.

It is important to note that system's Internet settings are combined with the configuration settings, with the latter taking precedence.

The following example sets the default proxy server address to `http://proxyserver`, indicates that the proxy should not be used for local addresses, and specifies that all requests to servers located in the `contoso.com` domain should bypass the proxy.

XML

```
<configuration>
  <system.net>
    <defaultProxy>
      <proxy
        usesystemdefault = "false"
        proxyaddress = "http://proxyserver:80"
        bypassonlocal = "true"
      />
      <bypasslist>
        <add address="http://[a-z]+\contoso\com/" />
      </bypasslist>
    </defaultProxy>
  </system.net>
</configuration>
```

Use the [<connectionManagement> Element \(Network Settings\)](#) element to configure the number of persistent connections that can be made to a specific server or to all other servers. The following example configures the application to use two persistent connections to the server `www.contoso.com`, four persistent connections to the server with the IP address `192.168.1.2`, and one persistent connection to all other servers.

XML

```
<configuration>
  <system.net>
    <connectionManagement>
      <add address="http://www.contoso.com" maxconnection="2" />
      <add address="192.168.1.2" maxconnection="4" />
      <add address="*" maxconnection="1" />
    </connectionManagement>
  </system.net>
</configuration>
```

Custom authentication modules are configured with the [`<authenticationModules>` Element \(Network Settings\)](#) element. Custom authentication modules must implement the [`IAuthenticationModule`](#) interface.

The following example configures a custom authentication module.

XML

```
<configuration>
  <system.net>
    <authenticationModules>
      <add type="MyAuthModule, MyAuthModule.dll" />
    </authenticationModules>
  </system.net>
</configuration>
```

You can use the [`<webRequestModules>` Element \(Network Settings\)](#) element to configure your application to use custom protocol-specific modules to request information from Internet resources. The specified modules must implement the [`IWebRequestCreate`](#) interface. You can override the default HTTP, HTTPS, and file request modules by specifying your custom module in the configuration file, as in the following example.

XML

```
<configuration>
  <system.net>
    <webRequestModules>
      <add
        prefix="HTTP"
        type = "MyHttpRequest.dll, MyHttpRequestCreator"
      />
    </webRequestModules>
  </system.net>
</configuration>
```

See also

- [Network programming in .NET](#)
- [Network Settings Schema](#)
- [<system.Net> Element \(Network Settings\)](#)

Configure apps by using configuration files

Article • 10/24/2023

ⓘ Note

This article is specific to .NET Framework. It doesn't apply to newer implementations of .NET, including .NET 6 and later versions.

.NET Framework gives developers and administrators control and flexibility over the way applications run through *configuration files*. Configuration files are XML files that can be changed as needed. An administrator can control which protected resources an application can access, which versions of assemblies an application will use, and where remote applications and objects are located. Developers can put settings in configuration files, eliminating the need to recompile an application every time a setting changes. This section describes what can be configured and why configuring an application might be useful.

ⓘ Note

Managed code can use the classes in the [System.Configuration](#) namespace to read settings from the configuration files, but not to write settings to those files.

This article describes the syntax of configuration files and provides information about the three types of configuration files: machine, application, and security.

Configuration file format

Configuration files contain elements, which are logical data structures that set configuration information. Within a configuration file, you use tags to mark the beginning and end of an element. For example, the `<runtime>` element consists of `<runtime> child elements </runtime>`. An empty element would be written as `<runtime/>` or `<runtime></runtime>`.

As with all XML files, the syntax in configuration files is case-sensitive.

You specify configuration settings using predefined attributes, which are name/value pairs inside an element's start tag. The following example specifies two attributes

(`version` and `href`) for the `<codeBase>` element, which specifies where the runtime can locate an assembly (for more information, see [Specifying an Assembly's Location](#)).

XML

```
<codeBase version="2.0.0.0"  
        href="http://www.litwareinc.com/myAssembly.dll"/>
```

Machine configuration files

The machine configuration file, *Machine.config*, contains settings that apply to an entire computer. This file is located in the `%runtime install path%\Config` directory.

Machine.config contains configuration settings for machine-wide assembly binding, built-in [remoting channels](#), and ASP.NET.

The configuration system first looks in the machine configuration file for the [`<appSettings>` element](#) and other configuration sections that a developer might define. It then looks in the application configuration file. To keep the machine configuration file manageable, it is best to put these settings in the application configuration file. However, putting the settings in the machine configuration file can make your system more maintainable. For example, if you have a third-party component that both your client and server application uses, it is easier to put the settings for that component in one place. In this case, the machine configuration file is the appropriate place for the settings, so you don't have the same settings in two different files.

 **Note**

Deploying an application using XCOPY will not copy the settings in the machine configuration file.

For more information about how the common language runtime uses the machine configuration file for assembly binding, see [How the Runtime Locates Assemblies](#).

Application configuration files

An application configuration file contains settings that are specific to an app. This file includes configuration settings that the common language runtime reads (such as assembly binding policy, remoting objects, and so on), and settings that the app can read.

The name and location of the application configuration file depend on the app's host, which can be one of the following:

- Executable-hosted app.

These apps have two configuration files: a source configuration file, which is modified by the developer during development, and an output file that's distributed with the app.

By default, the name of the source configuration file is *App.config*. When you create a .NET Framework project in Visual Studio, an *App.config* file is automatically added to the project. You can also add a file manually by selecting **File > New File**. Place the *App.config* file in the project directory and set its **Copy To Output Directory** property to **Copy always** or **Copy if newer**.

To create the output configuration file that's deployed with the app, Visual Studio copies the source configuration file to the directory where the compiled assembly is placed. This file is named *<yourappname>.exe.config*. For example, an app named *myApp.exe* has an output configuration file named *myApp.exe.config*.

In some cases, Visual Studio may modify the output configuration file. For more information, see [Redirect versions at the app level](#) in [Redirecting assembly versions](#).

- ASP.NET-hosted app.

For more information about ASP.NET configuration files, see [ASP.NET Configuration Settings](#).

Security configuration files

Security configuration files contain information about the code group hierarchy and permission sets associated with a policy level. We strongly recommend that you use the [Code Access Security Policy tool \(Caspol.exe\)](#) to modify security policy to ensure that policy changes do not corrupt the security configuration files.

 **Note**

Starting with .NET Framework 4, the security configuration files are present only if security policy has been changed.

The security configuration files are in the following locations:

- Enterprise policy configuration file: %*runtime-install-path*%\Config\Enterprisesec.config
- Machine policy configuration file: %*runtime-install-path*%\Config\Security.config
- User policy configuration file: %USERPROFILE%\Application data\Microsoft\CLR security config\vxx.xx\Security.config

See also

- [Configuration File Schema](#)
- [Specifying an Assembly's Location](#)
- [Redirecting Assembly Versions](#)
- [ASP.NET Web Site Administration](#)
- [Security Policy Management](#)
- [Caspol.exe \(Code Access Security Policy Tool\)](#)
- [Assemblies in .NET](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Debugging, tracing, and profiling

Article • 06/04/2024

ⓘ Note

This article is specific to .NET Framework. It doesn't apply to newer implementations of .NET, including .NET 6 and later versions.

To debug a .NET Framework application, the compiler and runtime environment must be configured to enable a debugger to attach to the application and to produce both symbols and line maps, if possible, for the application and its corresponding common intermediate language (CIL). After a managed application has been debugged, it can be profiled to boost performance. Profiling evaluates and describes the lines of source code that generate the most frequently executed code, and how much time it takes to execute them.

.NET Framework applications are easily debugged by using Visual Studio, which handles many of the configuration details. If Visual Studio is not installed, you can examine and improve the performance of .NET Framework applications by using the debugging classes in the .NET Framework [System.Diagnostics](#) namespace. This namespace includes the [Trace](#), [Debug](#), and [TraceSource](#) classes for tracing execution flow, and the [Process](#), [EventLog](#), and [PerformanceCounter](#) classes for profiling code.

In this section

[Enabling JIT-Attach Debugging](#)

Shows how to configure the registry to JIT-attach a debug engine to a .NET Framework application.

[Making an Image Easier to Debug](#)

Shows how to turn JIT tracking on and optimization off to make an assembly easier to debug.

[Tracing and Instrumenting Applications](#)

Describes how to monitor the execution of your application while it is running, and how to instrument it to display how well it is performing or whether something has gone wrong.

[Diagnosing Errors with Managed Debugging Assistants](#)

Describes managed debugging assistants (MDAs), which are debugging aids that work

in conjunction with the common language runtime (CLR) to provide information on runtime state.

[Enhancing Debugging with the Debugger Display Attributes](#)

Describes how the developer of a type can specify what that type will look like when it is displayed in a debugger.

[Runtime Profiling](#)

Learn how to gather information about application performance.

[Performance Counters](#)

Describes the counters that you can use to track the performance of an application.

Related sections

[Debug ASP.NET or ASP.NET Core apps in Visual Studio](#) Provides prerequisites and instructions for how to debug an ASP.NET application during development or after deployment.

[Development Guide](#) Provides a guide to all key technology areas and tasks for application development, including creating, configuring, debugging, securing, and deploying your application, and information about dynamic programming, interoperability, extensibility, memory management, and threading.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Deploy .NET Framework and applications

Article • 05/26/2022

ⓘ Note

This article is specific to .NET Framework. It doesn't apply to newer implementations of .NET, including .NET 6 and later versions.

This article helps you get started deploying .NET Framework with your application. Most of the information is intended for developers, OEMs, and enterprise administrators. Users who want to install .NET Framework on their computers should read [Install .NET Framework](#).

Key Deployment Resources

Use the following links to other MSDN topics for specific information about deploying and servicing the .NET Framework.

Setup and deployment

- General installer and deployment information:
 - Installer options:
 - [Web installer](#)
 - [Offline installer](#)
 - Installation modes:
 - [Silent installation](#)
 - [Displaying a UI](#)
 - [Reducing system restarts during .NET Framework 4.5 installations](#)
 - [Troubleshoot blocked .NET Framework installations and uninstallations](#)
- Deploying the .NET Framework with a client application (for developers):
 - [Using InstallShield in a setup and deployment project](#)

- [Using a Visual Studio ClickOnce application](#)
- [Creating a WiX installation package](#)
- [Using a custom installer](#)
- [Additional information](#) for developers
- Deploying the .NET Framework (for OEMs and administrators):
 - [Windows Assessment and Deployment Kit \(ADK\)](#)
 - [Administrator's guide](#)

Servicing

- For general information, see the [.NET Framework blog ↗](#).
- [Detecting versions](#)
- [Detecting service packs and updates](#)

Features That Simplify Deployment

The .NET Framework provides a number of basic features that make it easier to deploy your applications:

- No-impact applications.

This feature provides application isolation and eliminates DLL conflicts. By default, components do not affect other applications.

- Private components by default.

By default, components are deployed to the application directory and are visible only to the containing application.

- Controlled code sharing.

Code sharing requires you to explicitly make code available for sharing instead of being the default behavior.

- Side-by-side versioning.

Multiple versions of a component or application can coexist, you can choose which versions to use, and the common language runtime enforces versioning policy.

- XCOPY deployment and replication.

Self-described and self-contained components and applications can be deployed without registry entries or dependencies.

- On-the-fly updates.

Administrators can use hosts, such as ASP.NET, to update program DLLs, even on remote computers.

- Integration with the Windows Installer.

Advertisement, publishing, repair, and install-on-demand are all available when deploying your application.

- Enterprise deployment.

This feature provides easy software distribution, including using Active Directory.

- Downloading and caching.

Incremental downloads keep downloads smaller, and components can be isolated for use only by the application for low-impact deployment.

- Partially trusted code.

Identity is based on the code instead of the user, and no certificate dialog boxes appear.

Packaging and Distributing .NET Framework Applications

Some of the packaging and deployment information for the .NET Framework is described in other sections of the documentation. Those sections provide information about the self-describing units called [assemblies](#), which require no registry entries, [strong-named assemblies](#), which ensure name uniqueness and prevent name spoofing, and [assembly versioning](#), which addresses many of the problems associated with DLL conflicts. The following sections provide information about packaging and distributing .NET Framework applications.

Packaging

The .NET Framework provides the following options for packaging applications:

- As a single assembly or as a collection of assemblies.

With this option, you simply use the .dll or .exe files as they were built.

- As cabinet (CAB) files.

With this option, you compress files into .cab files to make distribution or download less time consuming.

- As a Windows Installer package or in other installer formats.

With this option, you create .msi files for use with the Windows Installer, or you package your application for use with some other installer.

Distribution

The .NET Framework provides the following options for distributing applications:

- Use XCOPY or FTP.

Because common language runtime applications are self-describing and require no registry entries, you can use XCOPY or FTP to simply copy the application to an appropriate directory. The application can then be run from that directory.

- Use code download.

If you are distributing your application over the Internet or through a corporate intranet, you can simply download the code to a computer and run the application there.

- Use an installer program such as Windows Installer 2.0.

Windows Installer 2.0 can install, repair, or remove .NET Framework assemblies in the global assembly cache and in private directories.

Installation Location

To determine where to deploy your application's assemblies so they can be found by the runtime, see [How the Runtime Locates Assemblies](#).

Security considerations can also affect how you deploy your application. Security permissions are granted to managed code according to where the code is located. Deploying an application or component to a location where it receives little trust, such as the internet, limits what the application or component can do.

Related Topics

 Expand table

Title	Description
How the Runtime Locates Assemblies	Describes how the common language runtime determines which assembly to use to fulfill a binding request.
Best Practices for Assembly Loading	Discusses ways to avoid problems of type identity that can lead to <code>InvalidCastException</code> , <code>MissingMethodException</code> , and other errors.
Reducing System Restarts During .NET Framework 4.5 Installations	Describes the Restart Manager, which prevents reboots whenever possible, and explains how applications that install the .NET Framework can take advantage of it.
Deployment Guide for Administrators	Explains how a system administrator can deploy the .NET Framework and its system dependencies across a network by using Microsoft Endpoint Configuration Manager.
Deployment Guide for Developers	Explains how developers can install .NET Framework on their users' computers with their applications.
Deploying Applications, Services, and Components	Discusses deployment options in Visual Studio, including instructions for publishing an application using the ClickOnce and Windows Installer technologies.
Publishing ClickOnce Applications	Describes how to package a Windows Forms application and deploy it with ClickOnce to client computers on a network.
Package and Deploy resources	Describes the hub and spoke model that the .NET Framework uses to package and deploy resources; covers resource naming conventions, fallback process, and packaging alternatives.
Deploying an Interop Application	Explains how to ship and install interop applications, which typically include a .NET Framework client assembly, one or more interop assemblies representing distinct COM type libraries, and one or more registered COM components.
How to: Get Progress from the .NET Framework 4.5 Installer	Describes how to silently launch and track the .NET Framework setup process while showing your own view of the setup progress.

See also

- [Development Guide](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

.NET Framework performance

Article • 05/17/2022

If you want to create apps with great performance, you should design and plan for performance just as you would design any other feature of your app. You can use the tools provided by Microsoft to measure your app's performance, and, if needed, make improvements to memory use, code throughput, and responsiveness. This topic lists the performance analysis tools that Microsoft provides, and provides links to other topics that cover performance for specific areas of app development.

Designing and planning for performance

If you want a great performing app, you must design performance into your app just as you would design any other feature. You should determine the performance-critical scenarios in your app, set performance goals, and measure performance for these app scenarios early and often. Because each app is different and has different performance-critical execution paths, determining those paths early and focusing your efforts enable you to maximize your productivity.

You don't have to be completely familiar with your target platform to create a high-performance app. However, you should develop an understanding of which parts of your target platform are costly in terms of performance. You can do this by measuring performance early in your development process.

To determine the areas that are crucial to performance and to establish your performance goals, always consider the user experience. Startup time and responsiveness are two key areas that will affect the user's perception of your app. If your app uses a lot of memory, it may appear sluggish to the user or affect other apps running on the system, or, in some cases, it could fail the Windows Store or Windows Phone Store submission process. Also, if you determine which parts of your code execute more frequently, you can make sure that these portions of your code are well optimized.

Analyzing performance

As part of your overall development plan, set points during development where you will measure the performance of your app and compare the results with the goals you set previously. Measure your app in the environment and hardware that you expect your users to have. By analyzing your app's performance early and often you can change architectural decisions that would be costly and expensive to fix later in the

development cycle. The following sections describe performance tools you can use to analyze your apps and discuss event tracing, which is used by these tools.

Performance tools

Here are some of the performance tools you can use with your .NET Framework apps.

[Expand table](#)

Tool	Description
Visual Studio Performance Analysis	Use to analyze the CPU usage of your .NET Framework apps that will be deployed to computers that are running the Windows operating system. This tool is available from the Debug menu in Visual Studio after you open a project. For more information, see Performance Explorer . Note: Use Windows Phone Application Analysis (see next row) when targeting Windows Phone.
Windows Phone Application Analysis	Use to analyze the CPU and memory, network data transfer rate, app responsiveness, and battery consumption in your Windows Phone apps. This tool is available from the Debug menu for a Windows Phone project in Visual Studio after you install the Windows Phone SDK . For more information, see App profiling for Windows Phone 8 .
PerfView ↗	Use to identify CPU and memory-related performance issues. This tool uses event tracing for Windows (ETW) and CLR profiling APIs to provide advanced memory and CPU investigations as well as information about garbage collection and JIT compilation. For more information about how to use PerfView, see the blog posts .
Windows Performance Analyzer ↗	Use to determine overall system performance such as your app's memory and storage use when multiple apps are running on the same computer. This tool is available from the download center as part of the Windows Assessment and Deployment Kit (ADK) for Windows 8. For more information, see Windows Performance Analyzer .

Event tracing for Windows (ETW)

ETW is a technique that lets you obtain diagnostic information about running code and is essential for many of the performance tools mentioned previously. ETW creates logs when particular events are raised by .NET Framework apps and Windows. With ETW, you can enable and disable logging dynamically, so that you can perform detailed tracing in a production environment without restarting your app. The .NET Framework offers support for ETW events, and ETW is used by many profiling and performance tools to generate performance data. These tools often enable and disable ETW events, so

familiarity with them is helpful. You can use specific ETW events to collect performance information about particular components of your app. For more information about ETW support in the .NET Framework, see [ETW Events in the Common Language Runtime](#) and [ETW Events in Task Parallel Library and PLINQ](#).

Performance by app type

Each type of .NET Framework app has its own best practices, considerations, and tools for evaluating performance. The following table links to performance topics for specific .NET Framework app types.

[+] [Expand table](#)

App type	See
.NET Framework apps for all platforms	Garbage Collection and Performance
	Performance Tips
Windows 8.x Store apps written in C++, C#, and Visual Basic	Performance best practices for Windows Store apps using C++, C#, and Visual Basic
Windows Presentation Foundation (WPF)	WPF Performance Suite
ASP.NET	ASP.NET Performance Overview

Related Topics

[+] [Expand table](#)

Title	Description
Caching in .NET Framework Applications	Describes techniques for caching data to improve performance in your app.
Lazy Initialization	Describes how to initialize objects as-needed to improve performance, particularly at app startup.
Reliability	Provides information about preventing asynchronous exceptions in a server environment.
Writing Large, Responsive .NET Framework Apps	Provides performance tips gathered from rewriting the C# and Visual Basic compilers in managed code, and includes several real examples from the C# compiler.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Dynamic language runtime overview

Article • 03/30/2024

The *dynamic language runtime* (DLR) is a runtime environment that adds a set of services for dynamic languages to the common language runtime (CLR). The DLR makes it easier to develop dynamic languages to run on .NET and to add dynamic features to statically typed languages.

Dynamic languages can identify the type of an object at run time, whereas in statically typed languages such as C# and Visual Basic (when you use `Option Explicit On`), you must specify object types at design time. Examples of dynamic languages are Lisp, Smalltalk, JavaScript, PHP, Ruby, Python, ColdFusion, Lua, Cobra, and Groovy.

Most dynamic languages provide the following advantages for developers:

- The ability to use a rapid feedback loop (REPL, or read-evaluate-print loop). This lets you enter several statements and immediately execute them to see the results.
- Support for both top-down development and more traditional bottom-up development. For example, when you use a top-down approach, you can call functions that aren't yet implemented and then add underlying implementations when you need them.
- Easier refactoring and code modifications, because you don't have to change static type declarations throughout the code.

Dynamic languages make excellent scripting languages. Customers can easily extend applications created by using dynamic languages with new commands and functionality. Dynamic languages are also frequently used for creating web sites and test harnesses, maintaining server farms, developing various utilities, and performing data transformations.

The purpose of the DLR is to enable a system of dynamic languages to run on .NET and give them .NET interoperability. The DLR adds dynamic objects to C# and Visual Basic to support dynamic behavior in these languages and enable their interoperation with dynamic languages.

The DLR also helps you create libraries that support dynamic operations. For example, if you have a library that uses XML or JavaScript Object Notation (JSON) objects, your objects can appear as dynamic objects to languages that use the DLR. This lets library users write syntactically simpler and more natural code for operating with objects and accessing object members.

For example, you might use the following code to increment a counter in XML in C#.

```
Scriptobj SetProperty("Count", ((int)GetProperty("Count")) + 1);
```

By using the DLR, you could use the following code instead for the same operation.

```
scriptobj.Count += 1;
```

Like the CLR, the DLR is a part of .NET. It's available for download on the [IronLanguages/dlr](#) repo on GitHub.

[IronPython](#) is an example of a language that was developed by using the DLR.

Primary DLR advantages

The DLR provides the following advantages.

Simplifies porting dynamic languages to .NET

The DLR allows language implementers to avoid creating lexical analyzers, parsers, semantic analyzers, code generators, and other tools that they traditionally had to create themselves. To use the DLR, a language needs to produce *expression trees*, which represent language-level code in a tree-shaped structure, runtime helper routines, and optional dynamic objects that implement the [IDynamicMetaObjectProvider](#) interface. The DLR and .NET automate a lot of code analysis and code generation tasks. This enables language implementers to concentrate on unique language features.

Enables dynamic features in statically typed languages

Existing .NET languages such as C# and Visual Basic can create dynamic objects and use them together with statically typed objects. For example, C# and Visual Basic can use dynamic objects for HTML, Document Object Model (DOM), and reflection.

Provides future benefits of the DLR and .NET

Languages implemented by using the DLR can benefit from future DLR and .NET improvements. For example, if .NET releases a new version that has an improved garbage collector or faster assembly loading time, languages implemented by using the DLR immediately get the same benefit. If the DLR adds optimizations such as better compilation, the performance also improves for all languages implemented by using the DLR.

Enables sharing of libraries and objects

The objects and libraries implemented in one language can be used by other languages. The DLR also enables interoperation between statically typed and dynamic languages. For example, C# can declare a dynamic object that uses a library that is written in a dynamic language. At the same time, dynamic languages can use libraries from the .NET Framework.

Provides fast dynamic dispatch and invocation

The DLR provides fast execution of dynamic operations by supporting advanced polymorphic caching. The DLR creates rules for binding operations that use objects to the necessary runtime implementations and then caches these rules to avoid resource-exhausting binding computations during successive executions of the same code on the same types of objects.

DLR architecture

The DLR adds a set of services to the CLR for better supporting dynamic languages. These services include the following:

- Expression trees. The DLR uses expression trees to represent language semantics. For this purpose, the DLR has extended LINQ expression trees to include control flow, assignment, and other language-modeling nodes. For more information, see [Expression Trees \(C#\)](#) or [Expression Trees \(Visual Basic\)](#).
- Call site caching. A *dynamic call site* is a place in the code where you perform an operation like `a + b` or `a.b()` on dynamic objects. The DLR caches the characteristics of `a` and `b` (usually the types of these objects) and information about the operation. If such an operation has been performed previously, the DLR retrieves all the necessary information from the cache for fast dispatch.
- Dynamic object interoperability. The DLR provides a set of classes and interfaces that represent dynamic objects and operations and can be used by language implementers and authors of dynamic libraries. These classes and interfaces include [IDynamicMetaObjectProvider](#), [DynamicMetaObject](#), [DynamicObject](#), and [ExpandoObject](#).

The DLR uses binders in call sites to communicate not only with .NET, but with other infrastructures and services, such as COM. Binders encapsulate a language's semantics and specify how to perform operations in a call site by using expression trees. This enables dynamic and statically typed languages that use the DLR to share libraries and gain access to all the technologies that the DLR supports.

DLR documentation

For more information about how to use the open source version of the DLR to add dynamic behavior to a language, or about how to enable the use of a dynamic language with .NET, see the documentation on the [IronLanguages/dlr](#) repo on GitHub.

See also

- [ExpandoObject](#)
- [DynamicObject](#)
- [Common Language Runtime](#)
- [Expression Trees \(C#\)](#)
- [Expression Trees \(Visual Basic\)](#)
- [Walkthrough: Create and Use Dynamic Objects](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Managed Extensibility Framework (MEF)

Article • 11/06/2021

This article provides an overview of the Managed Extensibility Framework that was introduced in .NET Framework 4.

What is MEF?

The Managed Extensibility Framework (MEF) is a library for creating lightweight and extensible applications. It allows application developers to discover and use extensions with no configuration required. It also lets extension developers easily encapsulate code and avoid fragile hard dependencies. MEF not only allows extensions to be reused within applications, but across applications as well.

The problem of extensibility

Imagine that you are the architect of a large application that must provide support for extensibility. Your application has to include a potentially large number of smaller components, and is responsible for creating and running them.

The simplest approach to the problem is to include the components as source code in your application, and call them directly from your code. This has a number of obvious drawbacks. Most importantly, you cannot add new components without modifying the source code, a restriction that might be acceptable in, for example, a Web application, but is unworkable in a client application. Equally problematic, you may not have access to the source code for the components, because they might be developed by third parties, and for the same reason you cannot allow them to access yours.

A slightly more sophisticated approach would be to provide an extension point or interface, to permit decoupling between the application and its components. Under this model, you might provide an interface that a component can implement, and an API to enable it to interact with your application. This solves the problem of requiring source code access, but it still has its own difficulties.

Because the application lacks any capacity for discovering components on its own, it must still be explicitly told which components are available and should be loaded. This is typically accomplished by explicitly registering the available components in a configuration file. This means that assuring that the components are correct becomes a maintenance issue, particularly if it is the end user and not the developer who is expected to do the updating.

In addition, components are incapable of communicating with one another, except through the rigidly defined channels of the application itself. If the application architect has not anticipated the need for a particular communication, it is usually impossible.

Finally, the component developers must accept a hard dependency on what assembly contains the interface they implement. This makes it difficult for a component to be used in more than one application, and can also create problems when you create a test framework for components.

What MEF provides

Instead of this explicit registration of available components, MEF provides a way to discover them implicitly, via *composition*. A MEF component, called a *part*, declaratively specifies both its dependencies (known as *imports*) and what capabilities (known as *exports*) it makes available. When a part is created, the MEF composition engine satisfies its imports with what is available from other parts.

This approach solves the problems discussed in the previous section. Because MEF parts declaratively specify their capabilities, they are discoverable at run time, which means an application can make use of parts without either hard-coded references or fragile configuration files. MEF allows applications to discover and examine parts by their metadata, without instantiating them or even loading their assemblies. As a result, there is no need to carefully specify when and how extensions should be loaded.

In addition to its provided exports, a part can specify its imports, which will be filled by other parts. This makes communication among parts not only possible, but easy, and allows for good factoring of code. For example, services common to many components can be factored into a separate part and easily modified or replaced.

Because the MEF model requires no hard dependency on a particular application assembly, it allows extensions to be reused from application to application. This also makes it easy to develop a test harness, independent of the application, to test extension components.

An extensible application written by using MEF declares an import that can be filled by extension components, and may also declare exports in order to expose application services to extensions. Each extension component declares an export, and may also declare imports. In this way, extension components themselves are automatically extensible.

Where MEF is available

MEF is an integral part of the .NET Framework 4, and is available wherever the .NET Framework is used. You can use MEF in your client applications, whether they use Windows Forms, WPF, or any other technology, or in server applications that use ASP.NET.

MEF and MAF

Previous versions of the .NET Framework introduced the Managed Add-in Framework (MAF), designed to allow applications to isolate and manage extensions. The focus of MAF is slightly higher-level than MEF, concentrating on extension isolation and assembly loading and unloading, while MEF's focus is on discoverability, extensibility, and portability. The two frameworks interoperate smoothly, and a single application can take advantage of both.

SimpleCalculator: An example application

The simplest way to see what MEF can do is to build a simple MEF application. In this example, you build a very simple calculator named SimpleCalculator. The goal of SimpleCalculator is to create a console application that accepts basic arithmetic commands, in the form "5+3" or "6-2", and returns the correct answers. Using MEF, you'll be able to add new operators without changing the application code.

To download the complete code for this example, see the [SimpleCalculator sample \(Visual Basic\)](#).

ⓘ Note

The purpose of SimpleCalculator is to demonstrate the concepts and syntax of MEF, rather than to necessarily provide a realistic scenario for its use. Many of the applications that would benefit most from the power of MEF are more complex than SimpleCalculator. For more extensive examples, see the [Managed Extensibility Framework](#) on GitHub.

- To start, in Visual Studio, create a new Console Application project and name it `SimpleCalculator`.
- Add a reference to the `System.ComponentModel.Composition` assembly, where MEF resides.

- Open *Module1.vb* or *Program.cs* and add `Imports` or `using` directives for `System.ComponentModel.Composition` and `System.ComponentModel.Composition.Hosting`. These two namespaces contain MEF types you will need to develop an extensible application.
- If you're using Visual Basic, add the `Public` keyword to the line that declares the `Module1` module.

Composition container and catalogs

The core of the MEF composition model is the *composition container*, which contains all the parts available and performs composition. Composition is the matching up of imports to exports. The most common type of composition container is [CompositionContainer](#), and you'll use this for SimpleCalculator.

If you're using Visual Basic, add a public class named `Program` in *Module1.vb*.

Add the following line to the `Program` class in *Module1.vb* or *Program.cs*:

```
C#
private CompositionContainer _container;
```

In order to discover the parts available to it, the composition containers makes use of a *catalog*. A catalog is an object that makes available parts discovered from some source. MEF provides catalogs to discover parts from a provided type, an assembly, or a directory. Application developers can easily create new catalogs to discover parts from other sources, such as a Web service.

Add the following constructor to the `Program` class:

```
C#
private Program()
{
    try
    {
        // An aggregate catalog that combines multiple catalogs.
        var catalog = new AggregateCatalog();
        // Adds all the parts found in the same assembly as the Program
        // class.
        catalog.Catalogs.Add(new AssemblyCatalog(typeof(Program).Assembly));

        // Create the CompositionContainer with the parts in the catalog.
        _container = new CompositionContainer(catalog);
```

```
        _container.ComposeParts(this);
    }
    catch (CompositionException compositionException)
    {
        Console.WriteLine(compositionException.ToString());
    }
}
```

The call to `ComposeParts` tells the composition container to compose a specific set of parts, in this case the current instance of `Program`. At this point, however, nothing will happen, since `Program` has no imports to fill.

Imports and Exports with attributes

First, you have `Program` import a calculator. This allows the separation of user interface concerns, such as the console input and output that will go into `Program`, from the logic of the calculator.

Add the following code to the `Program` class:

```
C#
[Import(typeof(ICalculator))]
public ICalculator calculator;
```

Notice that the declaration of the `calculator` object is not unusual, but that it is decorated with the `ImportAttribute` attribute. This attribute declares something to be an import; that is, it will be filled by the composition engine when the object is composed.

Every import has a *contract*, which determines what exports it will be matched with. The contract can be an explicitly specified string, or it can be automatically generated by MEF from a given type, in this case the interface `ICalculator`. Any export declared with a matching contract will fulfill this import. Note that while the type of the `calculator` object is in fact `ICalculator`, this is not required. The contract is independent from the type of the importing object. (In this case, you could leave out the `typeof(ICalculator)`. MEF will automatically assume the contract to be based on the type of the import unless you specify it explicitly.)

Add this very simple interface to the module or `SimpleCalculator` namespace:

```
C#
public interface ICalculator
{
```

```
        string Calculate(string input);
    }
```

Now that you have defined `ICalculator`, you need a class that implements it. Add the following class to the module or `SimpleCalculator` namespace:

C#

```
[Export(typeof(ICalculator))]
class MySimpleCalculator : ICalculator
{
}
```

Here is the export that will match the import in `Program`. In order for the export to match the import, the export must have the same contract. Exporting under a contract based on `typeof(MySimpleCalculator)` would produce a mismatch, and the import would not be filled; the contract needs to match exactly.

Since the composition container will be populated with all the parts available in this assembly, the `MySimpleCalculator` part will be available. When the constructor for `Program` performs composition on the `Program` object, its import will be filled with a `MySimpleCalculator` object, which will be created for that purpose.

The user interface layer (`Program`) does not need to know anything else. You can therefore fill in the rest of the user interface logic in the `Main` method.

Add the following code to the `Main` method:

C#

```
static void Main(string[] args)
{
    // Composition is performed in the constructor.
    var p = new Program();
    Console.WriteLine("Enter Command:");
    while (true)
    {
        string s = Console.ReadLine();
        Console.WriteLine(p.calculator.Calculate(s));
    }
}
```

This code simply reads a line of input and calls the `Calculate` function of `ICalculator` on the result, which it writes back to the console. That is all the code you need in

Program. All the rest of the work will happen in the parts.

Imports and ImportMany attributes

In order for SimpleCalculator to be extensible, it needs to import a list of operations. An ordinary `ImportAttribute` attribute is filled by one and only one `ExportAttribute`. If more than one is available, the composition engine produces an error. To create an import that can be filled by any number of exports, you can use the `ImportManyAttribute` attribute.

Add the following operations property to the `MySimpleCalculator` class:

C#

```
[ImportMany]  
IEnumerable<Lazy<IOperation, IOperationData>> operations;
```

`Lazy<T,TMetadata>` is a type provided by MEF to hold indirect references to exports. Here, in addition to the exported object itself, you also get *export metadata*, or information that describes the exported object. Each `Lazy<T,TMetadata>` contains an `IOperation` object, representing an actual operation, and an `IOperationData` object, representing its metadata.

Add the following simple interfaces to the module or `SimpleCalculator` namespace:

C#

```
public interface IOperation  
{  
    int Operate(int left, int right);  
}  
  
public interface IOperationData  
{  
    char Symbol { get; }  
}
```

In this case, the metadata for each operation is the symbol that represents that operation, such as +, -, *, and so on. To make the addition operation available, add the following class to the module or `SimpleCalculator` namespace:

C#

```
[Export(typeof(IOperation))]  
[ExportMetadata("Symbol", '+')]
```

```
class Add: IOperation
{
    public int Operate(int left, int right)
    {
        return left + right;
    }
}
```

The `ExportAttribute` attribute functions as it did before. The `ExportMetadataAttribute` attribute attaches metadata, in the form of a name-value pair, to that export. While the `Add` class implements `IOperation`, a class that implements `IOperationData` is not explicitly defined. Instead, a class is implicitly created by MEF with properties based on the names of the metadata provided. (This is one of several ways to access metadata in MEF.)

Composition in MEF is *recursive*. You explicitly composed the `Program` object, which imported an `ICalculator` that turned out to be of type `MySimpleCalculator`. `MySimpleCalculator`, in turn, imports a collection of `IOperation` objects, and that import will be filled when `MySimpleCalculator` is created, at the same time as the imports of `Program`. If the `Add` class declared a further import, that too would have to be filled, and so on. Any import left unfilled results in a composition error. (It is possible, however, to declare imports to be optional or to assign them default values.)

Calculator logic

With these parts in place, all that remains is the calculator logic itself. Add the following code in the `MySimpleCalculator` class to implement the `Calculate` method:

```
C#
public String Calculate(string input)
{
    int left;
    int right;
    char operation;
    // Finds the operator.
    int fn = FindFirstNonDigit(input);
    if (fn < 0) return "Could not parse command.";

    try
    {
        // Separate out the operands.
        left = int.Parse(input.Substring(0, fn));
        right = int.Parse(input.Substring(fn + 1));
    }
    catch
```

```

    {
        return "Could not parse command.";
    }

    operation = input[fn];

    foreach (Lazy<IOperation, IOperationData> i in operations)
    {
        if (i.Metadata.Symbol.Equals(operation))
        {
            return i.Value.Operate(left, right).ToString();
        }
    }
    return "Operation Not Found!";
}

```

The initial steps parse the input string into left and right operands and an operator character. In the `foreach` loop, every member of the `operations` collection is examined. These objects are of type `Lazy<T,TMetadata>`, and their metadata values and exported object can be accessed with the `Metadata` property and the `Value` property respectively. In this case, if the `symbol` property of the `IOperationData` object is discovered to be a match, the calculator calls the `Operate` method of the `IOperation` object and returns the result.

To complete the calculator, you also need a helper method that returns the position of the first non-digit character in a string. Add the following helper method to the `MySimpleCalculator` class:

```
C#
private int FindFirstNonDigit(string s)
{
    for (int i = 0; i < s.Length; i++)
    {
        if (!char.IsDigit(s[i])) return i;
    }
    return -1;
}
```

You should now be able to compile and run the project. In Visual Basic, make sure that you added the `Public` keyword to `Module1`. In the console window, type an addition operation, such as "5+3", and the calculator returns the results. Any other operator results in the "Operation Not Found!" message.

Extend SimpleCalculator using a new class

Now that the calculator works, adding a new operation is easy. Add the following class to the module or `SimpleCalculator` namespace:

```
C#  
  
[Export(typeof(IOperation))]  
[ExportMetadata("Symbol", "-")]  
class Subtract : IOperation  
{  
    public int Operate(int left, int right)  
    {  
        return left - right;  
    }  
}
```

Compile and run the project. Type a subtraction operation, such as "5-3". The calculator now supports subtraction as well as addition.

Extend SimpleCalculator using a new assembly

Adding classes to the source code is simple enough, but MEF provides the ability to look outside an application's own source for parts. To demonstrate this, you will need to modify SimpleCalculator to search a directory, as well as its own assembly, for parts, by adding a [DirectoryCatalog](#).

Add a new directory named `Extensions` to the SimpleCalculator project. Make sure to add it at the project level, and not at the solution level. Then add a new Class Library project to the solution, named `ExtendedOperations`. The new project will compile into a separate assembly.

Open the Project Properties Designer for the ExtendedOperations project and click the **Compile or Build** tab. Change the **Build output path** or **Output path** to point to the `Extensions` directory in the SimpleCalculator project directory (`..\SimpleCalculator\Extensions\`).

In `Module1.vb` or `Program.cs`, add the following line to the `Program` constructor:

```
C#  
  
catalog.Catalogs.Add(  
    new DirectoryCatalog(  
        "C:\\SimpleCalculator\\SimpleCalculator\\Extensions"));
```

Replace the example path with the path to your Extensions directory. (This absolute path is for debugging purposes only. In a production application, you would use a relative path.) The [DirectoryCatalog](#) will now add any parts found in any assemblies in the Extensions directory to the composition container.

In the `ExtendedOperations` project, add references to `SimpleCalculator` and `System.ComponentModel.Composition`. In the `ExtendedOperations` class file, add an `Imports` or a `using` directive for `System.ComponentModel.Composition`. In Visual Basic, also add an `Imports` statement for `SimpleCalculator`. Then add the following class to the `ExtendedOperations` class file:

C#

```
[Export(typeof(SimpleCalculator.IOperation))]
[ExportMetadata("Symbol", "%")]
public class Mod : SimpleCalculator.IOperation
{
    public int Operate(int left, int right)
    {
        return left % right;
    }
}
```

Note that in order for the contract to match, the [ExportAttribute](#) attribute must have the same type as the [ImportAttribute](#).

Compile and run the project. Test the new Mod (%) operator.

Conclusion

This topic covered the basic concepts of MEF.

- Parts, catalogs, and the composition container

Parts and the composition container are the basic building blocks of a MEF application. A part is any object that imports or exports a value, up to and including itself. A catalog provides a collection of parts from a particular source. The composition container uses the parts provided by a catalog to perform composition, the binding of imports to exports.

- Imports and exports

Imports and exports are the way by which components communicate. With an import, the component specifies a need for a particular value or object, and with

an export it specifies the availability of a value. Each import is matched with a list of exports by way of its contract.

Next steps

To download the complete code for this example, see the [SimpleCalculator sample \(Visual Basic\)](#).

For more information and code examples, see [Managed Extensibility Framework](#). For a list of the MEF types, see the [System.ComponentModel.Composition](#) namespace.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Interoperating with unmanaged code

Article • 03/11/2022

The .NET Framework promotes interaction with COM components, COM+ services, external type libraries, and many operating system services. Data types, method signatures, and error-handling mechanisms vary between managed and unmanaged object models. To simplify interoperation between .NET Framework components and unmanaged code and to ease the migration path, the common language runtime conceals from both clients and servers the differences in these object models.

Code that executes under the control of the runtime is called managed code. Conversely, code that runs outside the runtime is called unmanaged code. COM components, ActiveX interfaces, and Windows API functions are examples of unmanaged code.

In this section

[Exposing COM Components to the .NET Framework](#)

Describes how to use COM components from .NET Framework applications.

[Exposing .NET Framework Components to COM](#)

Describes how to use .NET Framework components from COM applications.

[Consuming Unmanaged DLL Functions](#)

Describes how to call unmanaged DLL functions using platform invoke.

[Interop Marshaling](#)

Describes marshalling for COM interop and platform invoke.

[How to: Map HRESULTs and Exceptions](#)

Describes the mapping between exceptions and HRESULTs.

[Type Equivalence and Embedded Interop Types](#)

Describes how type information for COM types is embedded in assemblies, and how the common language runtime determines the equivalence of embedded COM types.

[How to: Generate Primary Interop Assemblies Using Tlbimp.exe](#)

Describes how to produce primary interop assemblies using *Tlbimp.exe* (Type Library Importer).

[How to: Register Primary Interop Assemblies](#)

Describes how to register the primary interop assemblies before you can reference them

in your projects.

[Registration-Free COM Interop](#)

Describes how COM interop can activate components without using the Windows registry.

[How to: Configure .NET Framework-Based COM Components for Registration-Free Activation](#)

Describes how to create an application manifest and how to create and embed a component manifest.

Related sections

[COM Wrappers](#)

Describes the wrappers provided by COM interop.

Unmanaged API Reference

Article • 09/15/2021

This section includes information on unmanaged APIs that can be used by managed-code-related applications, such as runtime hosts, compilers, disassemblers, obfuscators, debuggers, and profilers.

In This Section

[Common Data Types](#) Lists the common data types that are used, particularly in the unmanaged profiling and debugging APIs.

[ALink](#) Describes the ALink API, which supports the creation of .NET Framework assemblies and unbound modules.

[Authenticode](#) Supports the Authenticode XrML license creation and verification module.

[Constants](#) Describes the constants that are defined in CorSym.idl.

[Custom Interface Attributes](#) Describes component object model (COM) custom interface attributes.

[Debugging](#) Describes the debugging API, which enables a debugger to debug code that runs in the common language runtime (CLR) environment.

[Diagnostics Symbol Store](#) Describes the diagnostics symbol store API, which enables a compiler to generate symbol information for use by a debugger.

[Fusion](#) Describes the fusion API, which enables a runtime host to access the properties of an application's resources in order to locate the correct versions of those resources for the application.

[Hosting](#) Describes the hosting API, which enables unmanaged hosts to integrate the CLR into their applications.

[Metadata](#) Describes the metadata API, which enables a client such as a compiler to generate or access a component's metadata without the types being loaded by the CLR.

[Profiling](#) Describes the profiling API, which enables a profiler to monitor a program's execution by the CLR.

[Strong Naming](#) Describes the strong naming API, which enables a client to administer strong name signing for assemblies.

[WMI and Performance Counters](#) Describes the APIs that wrap calls to Windows Management Instrumentation (WMI) libraries.

[Tlbexp Helper Functions](#) Describes the two helper functions and interface used by the Type Library Exporter (Tlbexp.exe) during the assembly-to-type-library conversion process.

.NET Framework tools

Article • 09/21/2022

The .NET Framework tools make it easier for you to create, deploy, and manage applications and components that target the .NET Framework.

Most of the .NET Framework tools described in this section are automatically installed with Visual Studio. To download Visual Studio, visit the [Visual Studio Downloads](#) page.

You can run all the tools from the command line, with the exception of the Assembly Cache Viewer (*Shfusion.dll*). You must access *Shfusion.dll* from File Explorer.

The best way to run the command-line tools is by using one of the developer shells that Visual Studio installs. These utilities enable you to run the tools easily, without having to navigate to the installation folder. For more information, see [Developer Command Prompt and Developer PowerShell](#).

ⓘ Note

Some tools are specific to either 32-bit computers or 64-bit computers. Be sure to run the appropriate version of the tool for your computer.

In this section

[Al.exe \(Assembly Linker\)](#)

Generates a file that has an assembly manifest from modules or resource files.

[Aximp.exe \(Windows Forms ActiveX Control Importer\)](#)

Converts type definitions in a COM type library for an ActiveX control into a Windows Forms control.

[Caspol.exe \(Code Access Security Policy Tool\)](#)

Enables you to view and configure security policy for the machine policy level, the user policy level, and the enterprise policy level. In .NET Framework 4 and later, this tool does not affect code access security (CAS) policy unless the `<legacyCasPolicy>` element is set to `true`.

[Cert2spc.exe \(Software Publisher Certificate Test Tool\)](#)

Creates a Software Publisher's Certificate (SPC) from one or more X.509 certificates. This tool is for testing purposes only.

[Certmgr.exe \(Certificate Manager Tool\)](#)

Manages certificates, certificate trust lists (CTLs), and certificate revocation lists (CRLs).

[Clrver.exe \(CLR Version Tool\)](#)

Reports all the installed versions of the common language runtime (CLR) on the computer.

[CorFlags.exe \(CorFlags Conversion Tool\)](#)

Lets you configure the CorFlags section of the header of a portable executable (PE) image.

[Fuslogvw.exe \(Assembly Binding Log Viewer\)](#)

Displays information about assembly binds to help you diagnose why the .NET Framework cannot locate an assembly at run time.

[Gacutil.exe \(Global Assembly Cache Tool\)](#)

Lets you view and manipulate the contents of the global assembly cache and download cache.

[Ilasm.exe \(IL Assembler\)](#)

Generates a portable executable (PE) file from intermediate language (IL). You can run the resulting executable to determine whether the IL performs as expected.

[Ildasm.exe \(IL Disassembler\)](#)

Takes a portable executable (PE) file that contains intermediate language (IL) code and creates a text file that can be input to the IL Assembler (Ilasm.exe).

[Installutil.exe \(Installer Tool\)](#)

Enables you to install and uninstall server resources by executing the installer components in a specified assembly. (Works with classes in the `System.Configuration.Install` namespace.)

[Lc.exe \(License Compiler\)](#)

Reads text files that contain licensing information and produces a `.licenses` file that can be embedded in a common language runtime executable as a resource.

[Mage.exe \(Manifest Generation and Editing Tool\)](#)

Lets you create, edit, and sign application and deployment manifests. As a command-line tool, `Mage.exe` can be run from both batch scripts and other Windows-based applications, including ASP.NET applications.

[MageUI.exe \(Manifest Generation and Editing Tool, Graphical Client\)](#)

Supports the same functionality as the command-line tool Mage.exe, but uses a

Windows-based user interface (UI). Supports the same functionality as the command-line tool *Mage.exe*, but uses a Windows-based user interface (UI).

[MDbg.exe \(.NET Framework Command-Line Debugger\)](#)

Helps tools vendors and application developers find and fix bugs in programs that target the .NET Framework common language runtime. This tool uses the runtime debugging API to provide debugging services.

[Mgmtclassgen.exe \(Management Strongly Typed Class Generator\)](#)

Enables you to generate an early-bound managed class for a specified Windows Management Instrumentation (WMI) class.

[Mpgo.exe \(Managed Profile Guided Optimization Tool\)](#)

Enables you to tune native image assemblies using common end-user scenarios. Mpgo.exe allows the generation and consumption of profile data for native image application assemblies (not the .NET Framework assemblies) using training scenarios selected by the application developer.

[Ngen.exe \(Native Image Generator\)](#)

Improves the performance of managed applications through the use of native images (files containing compiled processor-specific machine code). The runtime can use native images from the cache instead of using the just-in-time (JIT) compiler to compile the original assembly.

[Peverify.exe \(PEVerify Tool\)](#)

Helps you verify whether your common intermediate language (CIL) code and associated metadata meet type safety requirements.

[Regasm.exe \(Assembly Registration Tool\)](#)

Reads the metadata within an assembly and adds the necessary entries to the registry. This enables COM clients to appear as .NET Framework classes.

[Regsvcs.exe \(.NET Services Installation Tool\)](#)

Loads and registers an assembly, generates and installs a type library into a specified COM+ version 1.0 application, and configures services that you have added programmatically to a class.

[Resgen.exe \(Resource File Generator\)](#)

Converts text (.txt or .restext) files and XML-based resource format (.resx) files to common language runtime binary (.resources) files that can be embedded in a runtime binary executable or compiled into satellite assemblies.

[SecAnnotate.exe \(.NET Security Annotator Tool\)](#)

Identifies the `SecurityCritical` and `SecuritySafeCritical` portions of an assembly.

[SignTool.exe \(Sign Tool\)](#)

Digitally signs files, verifies signatures in files, and time-stamps files.

[Sn.exe \(Strong Name Tool\)](#)

Helps create assemblies with strong names. This tool provides options for key management, signature generation, and signature verification.

[SOS.dll \(SOS Debugging Extension\)](#)

Helps you debug managed programs in the WinDbg.exe debugger and in Visual Studio by providing information about the internal common language runtime environment.

[SqlMetal.exe \(Code Generation Tool\)](#)

Generates code and mapping for the LINQ to SQL component of the .NET Framework.

[Storeadm.exe \(Isolated Storage Tool\)](#)

Manages isolated storage; provides options for listing the user's stores and deleting them.

[Tlbexp.exe \(Type Library Exporter\)](#)

Generates a type library that describes the types that are defined in a common language runtime assembly.

[Tlbimp.exe \(Type Library Importer\)](#)

Converts the type definitions found in a COM type library into equivalent definitions in a common language runtime assembly.

[Winmdexp.exe \(Windows Runtime Metadata Export Tool\)](#)

Exports a .NET Framework assembly that is compiled as a *.winmdobj* file into a Windows Runtime component, which is packaged as a *.winmd* file that contains both Windows Runtime metadata and implementation information.

[Winres.exe \(Windows Forms Resource Editor\)](#)

Helps you localize user interface (UI) resources (*.resx* or *.resources* files) that are used by Windows Forms. You can translate strings, and then size, move, and hide controls to accommodate the localized strings.

Related sections

[WPF Tools](#) Includes tools such as the isXPS Conformance tool (*isXPS.exe*) and performance profiling tools.

[Windows Communication Foundation Tools](#)

Includes tools that make it easier for you to create, deploy, and manage Windows Communication Foundation (WCF) applications.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Al.exe (Assembly Linker)

Article • 07/20/2022

The Assembly Linker generates a file that has an assembly manifest from one or more files that are either modules or resource files. A module is an intermediate language (IL) file that does not have an assembly manifest.

ⓘ Note

Starting with Visual Studio 2008, both the C# and Visual Basic compilers automatically embed a Win32 manifest into the assembly. For more information, see [-win32manifest \(C# Compiler Options\)](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
al sources options
```

Parameters

You can specify one or more of the following `sources`.

[+] Expand table

Source	Description
<code>file [, target]</code>	Copies the contents of <code>file</code> (a module) to the file name specified by <code>target</code> . After copying, <i>Al.exe</i> compiles <code>target</code> into an assembly.
<code>/embed[resource]: file [, name [, private]]</code>	Embeds the resource specified by <code>file</code> in the image that contains the assembly manifest; <i>Al.exe</i> copies the contents of <code>file</code> into the portable executable (PE) image.
The <code>name</code> parameter is an internal identifier for the resource. By	

Source	Description
	<p>default, resources are public in the assembly (visible to other assemblies). Specifying <code>private</code> makes the resource not visible to other assemblies.</p> <p>If <code>file</code> is a .NET Framework resource file created, for example, by the Resource File Generator (Resgen.exe) or in the development environment, it can be accessed with members in the System.Resources. For more information, see ResourceManager. For all other resources, use the <code>GetManifestResource*</code> methods in the Assembly to access the resource at run time.</p> <p>If only resource files are passed to <code>Al.exe</code>, the output file is a satellite resource assembly.</p>
<p>/link[resource]: <code>file [, name [, target [, private]]]</code></p>	<p>Links a resource file to an assembly. The resource specified by <code>file</code> becomes part of the assembly; the file is not copied. The <code>file</code> parameter can be in any file format. For example, you can specify a native DLL as the <code>file</code> parameter. This will make the native DLL part of the assembly so that it can be installed into the global assembly cache and accessed from managed code in the assembly. You can also do this by using the /linkresource compiler option. For more information, see -linkresource (C# Compiler Options).</p> <p>The <code>name</code> parameter is an internal identifier for the resource. The <code>target</code> parameter specifies a path and file name into which <code>Al.exe</code> copies the <code>file</code>. After copying, <code>Al.exe</code> compiles <code>target</code> into an assembly. By default, resources are public in the assembly (visible to other assemblies). Specifying <code>private</code> makes the resource not visible to other assemblies.</p> <p>If <code>file</code> is a .NET Framework resource file created, for example, by the Resource File Generator (Resgen.exe) or in the development environment, it can be accessed with members in the System.Resources namespace. For more information, see ResourceManager. For all other resources, use the <code>GetManifestResource*</code> methods in the Assembly class to access the resource at run time.</p> <p>If only resource files are passed to <code>Al.exe</code>, the output file is a satellite resource assembly.</p>

You can specify the following `options`; you must specify `/out`.

[+] Expand table

Option	Description
<code>/algid:</code> <code>id</code>	<p>Specifies an algorithm to hash all files in a multifile assembly except the file that contains the assembly manifest. The default algorithm is CALG_SHA1. See ALG_ID in the Platform SDK documentation for other algorithms. For the first release of the .NET Framework, only CALG_SHA1 and CALG_MD5 are valid.</p> <p>The hash values are stored in the file table of the assembly manifest. At installation and load time, the assembly's files are checked against their hashes.</p> <p>You can also specify this option as a custom attribute (AssemblyAlgorithmIdAttribute) in the source code for any module.</p>
<code>/base[address]:</code> <code>addr</code>	<p>Specifies the address at which a DLL will be loaded on the user's computer at run time. Applications load faster if you specify the base address of the DLLs, instead of letting the operating system relocate the DLLs in the process space.</p>
<code>/bugreport:</code> <code>filename</code>	<p>Creates a file (<code>filename</code>) that contains information for reporting bugs.</p>
<code>/comp[any]:</code> <code>text</code>	<p>Specifies a string for the Company field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify <code>/win32res</code>, <code>text</code> appears in File Explorer as the <code>Company</code> property for the file. If you specify <code>/win32res</code>, the company information in the specified resource file appears as the <code>Company</code> property in File Explorer.</p> <p>If <code>text</code> is an empty string (""), the Win32 <code>Company</code> resource appears as a single space.</p> <p>If you specify <code>/win32res</code>, <code>/company</code> will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyCompanyAttribute) in the source code for any CIL module.</p>
<code>/config[uration]:</code> <code>text</code>	<p>Specifies a string for the Configuration field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If <code>text</code> is an empty string, the Win32 Configuration resource appears as a single space.</p>

Option	Description
	<p>You can also specify this option as a custom attribute (AssemblyConfigurationAttribute) in the source code for any CIL module.</p>
/copy[right]: <code>text</code>	<p>Specifies a string for the Copyright field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify /win32res, /copyright appears in File Explorer as the Win32 Copyright resource.</p> <p>If <code>text</code> is an empty string, the Win32 Copyright resource appears as a single space.</p> <p>If you specify /win32res, /copyright will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyCopyrightAttribute) in the source code for any CIL module.</p>
/culture: <code>text</code>	<p>Specifies the culture string to associate with the assembly. Valid values for cultures are those defined by the Internet Requests for Comments (RFC) document 1766 titled "Tags for the Identification of Languages."</p> <p>Place the string in double quotation marks (" ") if <code>text</code> contains a space. There is no default culture string. This string is available for viewing with reflection.</p> <p>For information about valid <code>text</code> strings, see the CultureInfo.</p> <p>You can also specify this option as a custom attribute (AssemblyCultureAttribute) in the source code for any CIL module.</p>
/delay[sign][+ or -]	<p>Specifies whether the assembly will be fully or partially signed. Use /delysign- if you want a fully signed assembly. Use /delysign+ if you only want to include the public key in the assembly.</p> <p>When you request a fully signed assembly, <i>AI.exe</i> hashes the file that contains the manifest (assembly metadata) and signs that hash with the private key. The resulting digital signature is stored in the file that contains the manifest. When an assembly is delay signed, <i>AI.exe</i> does not compute and store the signature, but just reserves space in the file so the signature can be added later.</p> <p>The default is /delysign-.</p> <p>The /delysign option has no effect unless used with /keyfile or /keyname.</p>

Option	Description
	<p>For example, using <code>/delaysign+</code> enables a tester to put the assembly in the global cache. After testing, you can fully sign the assembly by including the private key in the assembly.</p> <p>Note: Before using the Gacutil.exe (Global Assembly Cache Tool) to put a delay-signed assembly into the global cache, use the Sn.exe (Strong Name Tool) to register the assembly for verification skipping. For example, <code>Sn.exe -Vr delaySignedAssembly</code>. Use this only for development.</p> <p>You can also specify this option as a custom attribute (AssemblyDelaySignAttribute) in the source code for any CIL module.</p>
<code>/descr[ption]:</code> <code>text</code>	<p>Specifies a string for the Description field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify <code>/win32res</code>, <code>/description</code> appears in File Explorer as the Win32 Comments resource.</p> <p>If <code>text</code> is an empty string, the Win32 Comments resource appears as a single space.</p> <p>If you specify <code>/win32res</code>, <code>/description</code> will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (Description) in the source code for any CIL module.</p>
<code>/e[vidence]:</code> <code>file</code>	<p>Embeds <code>file</code> in the assembly with the resource name of Security.Evidence.</p> <p>You cannot use Security.Evidence for regular resources.</p>
<code>/fileversion:</code> <code>version</code>	<p>Specifies a string for the File Version field in the assembly. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify <code>/win32res</code>, <code>/fileversion</code> will be used as the Win32 File Version resource. If you do not specify <code>/fileversion</code>, the Win32 File Version resource will be populated by the Win32 Assembly Version resource.</p> <p>If <code>/win32res</code> is specified, <code>/fileversion</code> does not affect the Win32 resource.</p> <p>You can also specify this option as a custom attribute (AssemblyFileVersionAttribute) in the source code for any CIL module.</p>
<code>/flags:</code> <code>flags</code>	<p>Specifies a value for the <code>Flags</code> field in the assembly. Possible values for <code>flags</code>:</p>

Option	Description
0x0000	The assembly is side-by-side compatible.
0x0010	The assembly cannot execute with other versions if they are executing in the same application domain.
0x0020	The assembly cannot execute with other versions if they are executing in the same process.
0x0030	The assembly cannot execute with other versions if they are executing on the same computer.
	<p>You can also specify this option as a custom attribute (AssemblyFlagsAttribute) in the source code for any CIL module.</p>
/fullpaths	Causes <i>Al.exe</i> to use the absolute path for any files that are reported in an error message.
/help	Displays command syntax and options for the tool.
/keyf[ile]: filename	<p>Specifies a file (<code>filename</code>) that contains a key pair or just a public key to sign an assembly. The compiler inserts the public key in the assembly manifest and then signs the final assembly with the private key. See the Strong Name Tool (Sn.exe) for information about generating key files and installing key pairs into key containers.</p>
	<p>If you are using delayed signing, this file will usually have the public key but not the private key.</p>
	<p>The public key (of the key pair) information appears in the <code>.publickey</code> field of the assembly.</p>
	<p>You can also specify this option as a custom attribute (AssemblyKeyFileAttribute) in the source code for any CIL module.</p>
	<p>If both <code>/keyfile</code> and <code>/keyname</code> are specified (either by command-line option or by custom attribute) in the same compilation, <i>Al.exe</i> will first try the container specified with <code>/keyname</code>. If that succeeds, the assembly is signed with the information in the key container. If <i>Al.exe</i> does not find the key container, it will try the file specified with <code>/keyfile</code>. If that succeeds, the assembly is signed with the information in the key file and the key information will be installed in the key container (similar to the <code>-i</code> option in Sn.exe) so that on the next compilation, the <code>/keyname</code> option will be valid.</p>

Option	Description
<code>/keyn[ame]: text</code>	<p>Specifies a container that holds a key pair. This will sign the assembly (give it a strong name) by inserting a public key into the assembly manifest. <i>Al.exe</i> will then sign the final assembly with the private key.</p> <p>Use <i>Sn.exe</i> to generate a key pair.</p> <p>The key information appears in the <code>.publickey</code> field of the assembly.</p> <p>Place <code>text</code> in double quotation marks (" ") if there is an embedded space. You can also specify this option as a custom attribute (AssemblyKeyNameAttribute) in the source code for any CIL module.</p>
<code>/main: method</code>	<p>Specifies the fully qualified name (<code>class.method</code>) of the method to use as an entry point when converting a module to an executable file.</p>
<code>/nologo</code>	<p>Suppresses the banner, or logo, displayed at the command line when you invoke <i>Al.exe</i>.</p>
<code>/out: filename</code>	<p>Specifies the name of the file produced by <i>Al.exe</i>. This is a required option.</p>
<code>/platform: text</code>	<p>Limits which platform this code can run on; must be one of x86, Itanium, x64, anycpu (the default), or anycpu32bitpreferred.</p>
<code>/prod[uct]: text</code>	<p>Specifies a string for the Product field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify <code>/win32res</code>, <code>/product</code> appears in File Explorer as the Win32 Product Name resource.</p> <p>If <code>text</code> is an empty string, the Win32 Product Name resource appears as a single space.</p> <p>If you specify <code>/win32res</code>, <code>/product</code> will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyProductAttribute) in the source code for any CIL module.</p>
<code>/productv[ersion]: text</code>	<p>Specifies a string for the Product Version field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify <code>/win32res</code>, <code>/productversion</code> will be used as the Win32 Product Version resource. If you do not specify <code>/productversion</code>, the Win32</p>

Option	Description
	<p>Product Version resource will be populated by the Win32 File Version resource.</p> <p>If you specify /win32res, /productversion will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyInformationalVersionAttribute) in the source code for any CIL module.</p>
/t[arget]: <code>lib[rary] exe win[exe]</code>	<p>Specifies the file format of the output file: <code>lib[rary]</code> (code library), <code>exe</code> (console application), or <code>win[exe]</code> (Windows-based application). The default is <code>lib[rary]</code>.</p>
/template: <code>filename</code>	<p>Specifies the assembly, <code>filename</code>, from which to inherit all assembly metadata, except the culture field.</p> <p>An assembly that you create with /template will be a satellite assembly.</p>
/title: <code>text</code>	<p>Specifies a string for the Title field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify /win32res, /title appears in File Explorer as the Win32 Description resource, which is used by the shell as the friendly name of an application. It is also displayed on the Open With submenu of the shortcut menu for a file type for which there are multiple supporting applications.</p> <p>If <code>text</code> is an empty string, the Win32 Description resource appears as a single space.</p> <p>If you specify /win32res, /title will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyTitleAttribute) in the source code for any CIL module.</p>
/trade[mark]: <code>text</code>	<p>Specifies a string for the Trademark field in the assembly. Place the string in double quotation marks (" ") if <code>text</code> contains a space. This string is a custom attribute on the assembly and is available for viewing with reflection.</p> <p>If you do not specify /win32res, /trademark appears in File Explorer as the Win32 Trademark resource.</p> <p>If <code>text</code> is an empty string, the Win32 Trademark resource appears as a single space.</p>

Option	Description
	<p>If you specify <code>/win32res</code>, <code>/trademark</code> will not affect the Win32 resource information.</p> <p>You can also specify this option as a custom attribute (AssemblyTrademarkAttribute) in the source code for any CIL module.</p>
<code>/v[ersion]: version</code>	<p>Specifies version information for this assembly. The format of the version string is <code>major.minor.build.revision</code>. The default value is 0.</p>

Option	Description
	You can also specify this option as a custom attribute (AssemblyVersionAttribute) in the source code for any CIL module.
<code>/win32icon: filename</code>	Inserts an .ico file in the assembly. The .ico file gives the output file the desired appearance in File Explorer.
<code>/win32res: filename</code>	Inserts a Win32 resource (.res file) in the output file. A Win32 resource file can be created by using the Resource Compiler. The Resource Compiler is invoked when you compile a Visual C++ program; a .res file is created from the .rc file.
@filename	<p>Specifies a response file that contains <i>Al.exe</i> commands.</p> <p>Commands in the response file can appear one per line or on the same line, separated by one or more spaces.</p>
<code>/?</code>	Displays command syntax and options for the tool.

Remarks

All Visual Studio compilers produce assemblies. However, if you have one or more modules (metadata without a manifest), you can use *Al.exe* to create an assembly with the manifest in a separate file.

To install assemblies in the cache, remove assemblies from the cache, or list the contents of the cache, use the [Global Assembly Cache Tool \(Gacutil.exe\)](#).

Errors and Warnings

The following table lists the errors generated by *Al.exe*.

[] [Expand table](#)

Error	Description
al1001 Internal compiler error	<p>Try to determine whether <i>Al.exe</i> is failing because of its inability to parse unexpected syntax. Then, contact Microsoft Product Support Services.</p>
al1002 Out of memory	<p><i>Al.exe</i> ran out of memory and stopped. Increase the amount of available memory.</p>

Error	Description
al1003	<p>Compiler option 'option' must be followed by an argument</p> <p><i>Al.exe</i> expected an argument to be passed to a command-line option. For example, if you specify /algid:, you must pass an algorithm identifier.</p>
al1004	<p>Unexpected common language runtime initialization error — 'reason'</p> <p><i>Al.exe</i> reported an error with the installation of Visual Studio or the common language runtime for the specified reason.</p>
al1005	<p>File 'file' too big to open</p> <p>All files opened by <i>Al.exe</i> must be smaller than 4 gigabytes (GB).</p>
al1006	<p>Response file 'file' was already included</p> <p>The same response file was specified (@file) more than once on the command line. The response file can only be included once.</p>
al1007	<p>Error opening response file 'file' — 'reason'</p> <p><i>Al.exe</i> cannot open the specified response file for the specified reason.</p>
al1008	<p>Missing file specification for 'option' command-line option</p> <p><i>Al.exe</i> expected a file to be passed to a command-line option. For example, if you specify the /out option, you must specify a file.</p>
al1009	<p>Can't open 'file' for writing</p> <p><i>Al.exe</i> was unable to write to a file, such as the output assembly file. The disk might be full, the file might be read-only, or you might not have permissions to the file.</p>
al1010	<p>Command-line syntax error: Missing ':text' for 'option' option</p> <p><i>Al.exe</i> expected an argument to be passed to a command-line option. For example, if you specify the /title option, you must pass a string.</p>
al1011	<p>File 'file' is an executable file and cannot be opened as a text file</p> <p>A binary file was specified where a text file was expected. For example, this error occurs if a binary file is passed on the command line as a response file.</p>
al1012	<p>'value' is not a valid setting for option 'option'</p> <p>An unexpected value was passed to a command-line option. For example, this error occurs if you specify an invalid value to the /target option.</p>

Error	Description
al1013	Unrecognized command-line option: 'option' An invalid command-line option was specified.
al1014	Unexpected initialization error — 'reason' <i>Al.exe</i> detected a COM initialization failure. This might be caused by a lack of memory, but a more likely cause is the system DLL files. You should see a similar error if you run any Automation-aware or COM-aware program, such as Microsoft Visual Studio. Reinstall the operating system.
al1015	Unable to find messages file 'alinkui.dll' <i>Al.exe</i> requires <i>Alinkui.dll</i> . Make sure that this file is on your path. If necessary, copy it from the product CD.
al1016	No valid input files were specified <i>Al.exe</i> requires one or more input files that do not have assembly information.
al1017	No target file name was specified The required /out option specifying the target file name was missing.
al1018	Required file 'file' could not be loaded Certain DLL files cannot be loaded. Reinstall Visual Studio or the Windows SDK.
al1019	Metadata failure while creating assembly—reason Generation of the assembly was interrupted for the specified reason. For example, this error occurs if a file that you specify with the /win32res option is not found.
al1020	Ignoring included assembly 'file' An input file that contained an assembly was specified. <i>Al.exe</i> input files cannot contain assemblies.
al1021	'setting' : overriding previous setting A module had a value for a particular setting, possibly assigned through custom attributes, which was overridden with a value passed using an <i>Al.exe</i> command-line option.
al1022	Error reading embedded resource 'file'—reason <i>Al.exe</i> cannot read the file passed to the /embedresource option for the specified reason.

Error	Description
al1023	<p>Error embedding resource 'file'—reason</p> <p>The operating system cannot embed the resource file in the assembly for the specified reason.</p>
al1025	<p>ComType record 'record' points to an invalid file record 'record'</p> <p>Metadata in the input module is invalid. The tool that produced the module must be fixed.</p>
al1026	<p>The version specified 'version' is invalid</p> <p>See information about the /version option for valid formats.</p>
al1028	<p>Key file 'file' is missing the private key needed for signing</p> <p>A key file that contains only the public key was passed to the /keyfile option. Use the Strong Name Tool (Sn.exe) to generate a file that has both a public and private key, as shown in the following command.</p> <pre>sn -k keypair.snk.</pre>
al1029	<p>The key container name 'container' does not exist</p> <p>The value passed to the /keyname option is not a valid container. Use the Strong Name Tool (Sn.exe) to create a container.</p>
al1030	<p>The cryptographic service is not installed properly or does not have a suitable key provider</p> <p>You might have to reinstall the operating system or install some cryptographic utility that was used to create the key.</p>
al1031	<p>Error reading icon 'file'—reason</p> <p><i>Al.exe</i> cannot read the file that was passed to the /win32icon option for the specified reason</p>
al1032	<p>Error generating resources for 'file'—reason</p> <p><i>Al.exe</i> cannot create a file because of insufficient disk space or some other error. This error occurs when you specify the /win32icon option (which generates an .ico file) or do not specify the /win32res option (which generates a file that has resource information).</p> <p>If you cannot resolve the file generation problem, use /win32res, which specifies a file that can contain version or bitmap (icon) information.</p>
al1033	Assembly custom attribute 'attribute' was specified multiple times with different values

Error	Description
	Different values were passed to two occurrences of the same custom attribute in source modules that are specified as input to <i>Al.exe</i> .
al1034	Assembly 'file' cannot be copied or renamed
	While using the <i>Al.exe</i> syntax that enables you to both specify an input file and copy it, a name conflict arose that stopped the compiler. For example, this error occurs if you specify <code>input.dll,somename.dll /out:somename.dll</code> .
al1035	Libraries cannot have an entry point
	You cannot specify both the <code>/target:lib</code> option (the default) and the <code>/main</code> option.
al1036	Entry point required for executable applications
	When using the <code>/target:exe</code> or <code>/target:win</code> option, you must also specify the <code>/main</code> option.
al1037	Unable to find the entry point method 'main'
	<i>Al.exe</i> cannot find a <code>Main</code> method at the location specified by the <code>/main</code> option.
al1039	Initialization of global assembly cache manager failed—reason
	Reinstall Visual Studio or the Windows SDK.
al1040	Failed to install assembly into cache—reason
	Only signed assemblies can be installed into the cache. For more information, see Global Assembly Cache .
al1041	'method': cannot be the entry point because the signature or visibility is incorrect, or it is generic
	A method was specified with the <code>/main</code> option, but that method is not static, does not return <code>int</code> or <code>void</code> , was generic, or has invalid arguments.
al1042	'exe': EXEs cannot be added modules
	An <code>.exe</code> file that does not have an assembly was specified as an input file to <i>Al.exe</i> . <i>Al.exe</i> can only take <code>dll</code> files without assemblies as input files.
al1043	Manifest file name 'name' cannot be the same as any modules
	The name specified with the <code>/out</code> option cannot be the same as any one of the file names that are specified as input to <i>Al.exe</i> .
al1044	Error reading key file 'file'—reason

Error	Description
	An error occurred while opening or reading from a file specified with /keyfile or the AssemblyKeyFileAttribute .
al1045	Filename 'file' is too long or invalid A file name longer than 260 characters was passed to <i>Al.exe</i> . Choose a file name with fewer characters or a shorter path, or rename the file.
al1046	Resource identifier 'ID' has already been used in this assembly Two resources, embedded or linked, have the same identifier or name (the second argument). Remove or rename one of the conflicting resources.
al1047	Error importing file 'file'—reason A module file cannot be opened for the specified reason.
al1048	Error importing module 'module' of assembly 'assembly'—reason An error occurred when opening a nonmanifest file of a multifile assembly. This error is not emitted directly by <i>Al.exe</i> , but can be passed programmatically to a process that uses <i>Al.exe</i> .
al1049	Cannot auto-generate build and revision version numbers for dates before January 1, 2000 The system clock on your computer is set to a date earlier than January 1, 2000.
al1050	The feature you are using 'old feature' is no longer supported; please use 'new feature' instead A feature previously supported by <i>Al.exe</i> is now obsolete. Use the recommended feature instead.
al1051	Error emitting 'attribute' attribute—reason An assembly custom attribute was not processed by <i>Al.exe</i> for the specified reason.
al1052	File 'filename' is not an assembly The file specified with /template must contain assembly metadata. This error indicates that the file specified by /template did not contain an assembly.
al1053	The version 'version' specified for the 'option' is not in the normal 'major.minor.build.revision' format <i>Al.exe</i> detected ill-formed version information specified with the /fileversion or /productversion options.

Error	Description
al1054	<p>The version 'version' specified for the 'option' is not in the normal 'major.minor.build.revision' format</p> <p><i>Al.exe</i> detected ill-formed version information specified with the SatelliteContractVersionAttribute.</p>
al1055	<p>Referenced assembly 'filename' does not have a strong name</p> <p>This error is issued when you are building an assembly with a strong name and reference an assembly that does not have a strong name. To fix this, you must either regenerate your assembly with a strong name, or attach a strong name to the assembly by using <i>Sn.exe</i> (see the documentation for Sn.exe).</p> <p>A common occurrence of this error is when you are using COM objects by way of wrapper assemblies, such as when you add a reference to a COM module to a C# project by way of the Visual Studio IDE. To avoid the error, you can specify the strong name key file for COM wrapper assemblies in the Project Property "Wrapper Assembly Key File/Name"</p> <p>If you are creating the wrapper assembly through <i>tlbimp</i>, see the tlbimp documentation for information about how to assign a strong name to the wrapper assembly.</p> <p>If an assembly has a strong name, it can be installed in the global assembly cache. Consequently, referenced assemblies would also go into the global assembly cache. Only assemblies with strong names can go into the global assembly cache.</p>
al1056	<p>Referenced assembly 'filename' is a localized satellite assembly</p> <p>An assembly created by using the AssemblyCultureAttribute attribute was referenced in creating the current assembly. The AssemblyCultureAttribute attribute indicates the file is a localized satellite assembly and it is not appropriate to reference a satellite assembly. Reference the main parent assembly instead.</p>
al1057	<p>Executables cannot be localized, Culture should always be empty</p> <p>An assembly is being created by using <i>/target:exe</i> but <i>/culture</i> was specified. Assemblies in the .exe cannot have information in the Culture field.</p>
al1058	<p>'file' is an assembly and cannot be added as a module</p> <p>In a C++ compilation, <i>/assemblymodule</i> (linker option) was passed a file that contained an assembly.</p>
al1059	<p>Unknown error (code)</p> <p><i>Al.exe</i> received an unknown error code (code).</p> <p>Possible solutions include the following:</p>

Error	Description
	<p>Reinstall Visual Studio.</p> <p>Reinstall the Windows SDK.</p> <p>Check for missing files.</p> <p>Check for adequate disk space.</p> <p>Check for adequate memory.</p> <p>Stop other processes that might be accessing the files.</p> <p>Reboot your computer.</p>
al1060	<p>Cryptographic failure while creating hashes—reason</p> <p>An error occurred while creating the file hashes for a multifile assembly.</p>
al1061	<p>Cannot set option 'option' because 'reason'</p> <p>The value specified for this option is invalid for the specified reason.</p>
al1062	<p>Module 'module' was specified multiple times; it will only be included once</p> <p>This warning is generated when the same source, input, or module file is specified multiple times on the command line. Make sure that you specify the file name only once.</p>
al1063	<p>Public type 'type' is defined in multiple locations in this assembly: 'file1' and 'file2'</p> <p>The same type was found in more than one module in the assembly. Only one version of each type may be present in an assembly.</p>
al1064	<p>Cannot specify multiple /bugreport options.</p> <p>Only one /bugreport option is allowed.</p>
al1065	<p>File name 'File Name' is too long or invalid</p> <p>The specified file name is longer than the maximum allowed.</p>
al1066	<p>Character 'character' is not allowed on the command line or in response files</p> <p>An invalid character was found, either on the command line or in a file.</p>
al1067	<p>'filename' is a binary file instead of a text file</p> <p>The file is in binary format instead of text.</p>

Error	Description
al1068	<p>Module 'ModuleName' is already defined in this assembly. Each linked resource and module must have a unique file name.</p> <p>The module occurs more than once in this assembly.</p>
al1069	<p>Cannot create short file name 'filename' when a long file name with the same short file name already exists</p> <p>The current file has a name that is the short version of a file name that already exists. For example, compiling LongFileName.cs and then recompiling with the name LongFi~1.cs will cause a compiler error similar to this. If the compiler output files that have long names were deleted, but the analogous linker files remained, this error might occur.</p>
al1070	<p>Agnostic assembly cannot have a processor-specific module 'Module Name'</p> <p>If you are building using /platform:agnostic (or you don't specify /platform), an error will be generated if you try to add a module (using /addmodule) that is not agnostic. This is like trying to link an i386 obj file to an ia64 obj.</p> <p>The main source of non-agnostic modules is C++. If you are using /addmodule with a C++ module, you may have to modify your build scripts to specify the appropriate /platform setting.</p>
al1072	<p>Assembly and module 'Module Name' cannot target different processors</p> <p>You cannot link an assembly and a module that are targeted for different processors, because the result has to run on a single processor.</p>
al1073	<p>Referenced assembly 'assembly' targets a different processor</p> <p>You cannot link assemblies that are targeted for different processors, because the result has to run on a single processor.</p>
al1074	<p>Module name 'Module Name' stored in 'File Name' must match its file name</p> <p>This is required of the linker. To resolve this problem, make the two names match.</p>
al1075	<p>Delay signing was requested, but no key was given</p> <p>When an assembly is delay signed, the compiler does not compute and store the signature, but reserves space in the file so the signature can be added later.</p> <p>For example, using /delsign+ enables a tester to put the assembly in the global cache. After testing, you can fully sign the assembly by adding the private key to the assembly by using the Assembly Linker utility.</p>
al1076	Type 'type' is forwarded to multiple assemblies: 'assembly' and 'assembly'.

Error	Description
	A type can only be forwarded to one assembly.
al1077	Public type 'type' is defined in 'assembly' and forwarded to 'assembly'. There is a duplicate public type in the assembly being generated. One is a valid type definition and the other is a type forwarder.

Example

The following command creates an executable file *t2a.exe* with an assembly from the `t2.netmodule` module. The entry point is the `Main` method in `MyClass`.

Console

```
al t2.netmodule /target:exe /out:t2a.exe /main:MyClass.Main
```

See also

- [Tools](#)
- [Sn.exe \(Strong Name Tool\)](#)
- [Gacutil.exe \(Global Assembly Cache Tool\)](#)
- [Programming with Assemblies](#)
- [Developer command-line shells](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Aximp.exe (Windows Forms ActiveX Control Importer)

Article • 03/30/2023

The ActiveX Control Importer converts type definitions in a COM type library for an ActiveX control into a Windows Forms control.

Windows Forms can only host Windows Forms controls — that is, classes that are derived from [Control](#). Aximp.exe generates a wrapper class for an ActiveX control that can be hosted on a Windows Form. This allows you to use the same design-time support and programming methodology applicable to other Windows Forms controls.

To host the ActiveX control, you must generate a wrapper control that derives from [AxHost](#). This wrapper control contains an instance of the underlying ActiveX control. It knows how to communicate with the ActiveX control, but it appears as a Windows Forms control. This generated control hosts the ActiveX control and exposes its properties, methods, and events as those of the generated control.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
aximp [options]{file.dll | file.ocx}
```

Remarks

Argument	Description
<i>file</i>	The name of the source file that contains the ActiveX control to convert. The <i>file</i> argument must have the extension .dll or .ocx.

Option	Description
--------	-------------

Option	Description
/delaySign	Specifies to Aximp.exe to sign the resulting control using delayed signing. You must specify this option with either the /keycontainer:, /keyfile:, or /publickey: option. For more information on the delayed signing process, see Delay Signing an Assembly .
/help	Displays command syntax and options for the tool.
/keycontainer: <i>containerName</i>	Signs the resulting control with a strong name using the public/private key pair found in the key container specified by <i>containerName</i> .
/keyfile: <i>filename</i>	Signs the resulting control with a strong name using the publisher's official public/private key pair found in <i>filename</i> .
/nologo	Suppresses the Microsoft startup banner display.
/out: <i>filename</i>	Specifies the name of the assembly to create.
/publickey: <i>filename</i>	Signs the resulting control with a strong name using the public key found in the file specified by <i>filename</i> .
/rcw: <i>filename</i>	Uses the specified runtime callable wrapper instead of generating a new one. You may specify multiple instances. The current directory is used for relative paths. For more information, see Runtime Callable Wrapper .
/silent	Suppresses the display of success messages.
/source	Generates C# source code for the Windows Forms wrapper.
/verbose	Specifies verbose mode; displays additional progress information.
/?	Displays command syntax and options for the tool.

Aximp.exe converts an entire ActiveX Control type library at one time and produces a set of assemblies that contain the common language runtime metadata and control implementation for the types defined in the original type library. The generated files are named according to the following pattern:

Common language runtime proxy for COM types: *progid.dll*

Windows Forms proxy for ActiveX controls (where Ax signifies ActiveX): *Axprogid.dll*

Note

If the name of a member of the ActiveX control matches a name defined in the .NET Framework, Aximp.exe will prefix the member name with "Ctl" when it creates the AxHost derived class. For example, if your ActiveX control has a member named

"Layout," it is renamed "CtlLayout" in the AxHost derived class because the Layout event is defined within the .NET Framework.

You can examine these generated files with tools such as [Ildasm.exe \(IL Disassembler\)](#).

Using Aximp.exe to generate a .NET assembly for the ActiveX WebBrowser control (shdocvw.dll) is not supported.

When you run Aximp.exe over shdocvw.dll, it will always create another file named shdocvw.dll in the directory from which the tool is run. If you place this generated file in the Documents directory, it can cause problems for Windows Explorer. When the computer is rebooted, Windows looks in the Documents directory before the system32 directory to find a copy of shdocvw.dll. It will use the copy it finds in Documents and attempt to load the managed wrappers. Windows Explorer won't function properly because it relies on the rendering engine in the version of shdocvw.dll located in the system32 directory. If this problem occurs, delete the copy of shdocvw.dll in the Documents directory and reboot the computer.

Example

The following command generates MediaPlayer.dll and AxMediaPlayer.dll for the Media Player control `msdxm.ocx`.

Console

```
aximp c:\systemroot\system32\msdxm.ocx
```

See also

- [Tools](#)
- [Ildasm.exe \(IL Disassembler\)](#)

Caspol.exe (Code Access Security Policy Tool)

Article • 06/03/2022

The Code Access Security (CAS) Policy tool (Caspol.exe) enables users and administrators to modify security policy for the machine policy level, the user policy level, and the enterprise policy level.

ⓘ Important

Starting with .NET Framework 4, Caspol.exe does not affect CAS policy unless the **<legacyCasPolicy> element** is set to `true`. Any settings shown or modified by CasPol.exe will only affect applications that opt into using CAS policy.

ⓘ Note

Code Access Security (CAS) has been deprecated across all versions of .NET Framework and .NET. Recent versions of .NET do not honor CAS annotations and produce errors if CAS-related APIs are used. Developers should seek alternative means of accomplishing security tasks.

ⓘ Note

64-bit computers include both 64-bit and 32-bit versions of security policy. To ensure that your policy changes apply to both 32-bit and 64-bit applications, run both the 32-bit and 64-bit versions of Caspol.exe.

The Code Access Security Policy tool is automatically installed with .NET Framework and with Visual Studio. You can find Caspol.exe in
%windir%\Microsoft.NET\Framework\version on 32-bit systems or
%windir%\Microsoft.NET\Framework64\version on 64-bit systems. (For example, the location is %windir%\Microsoft.NET\Framework64\v4.030319\caspol.exe for .NET Framework 4 on a 64-bit system.) Multiple versions of the tool might be installed if your computer is running multiple versions of .NET Framework side by side. You can run the tool from the installation directory. However, we recommend that you use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#), which doesn't require you to navigate to the installation folder.

At the command prompt, type the following:

Syntax

Console

```
caspol [options]
```

Parameters

[] [Expand table](#)

Option	Description
-addfulltrust <i>assembly_file</i> or -af <i>assembly_file</i>	Adds an assembly that implements a custom security object (such as a custom permission or a custom membership condition) to the full trust assembly list for a specific policy level. The <i>assembly_file</i> argument specifies the assembly to add. This file must be signed with a strong name . You can sign an assembly with a strong name using the Strong Name Tool (Sn.exe) .
-addgroup <i>{parent_label parent_name} {mship pset_name} [flags]</i> or -ag <i>{parent_label}</i>	Whenever a permission set containing a custom permission is added to policy, the assembly implementing the custom permission must be added to the full trust list for that policy level. Assemblies that implement custom security objects (such as custom code groups or membership conditions) used in a security policy (such as the machine policy) should always be added to the full trust assembly list. Caution: If the assembly implementing the custom security object references other assemblies, you must first add the referenced assemblies to the full trust assembly list. Custom security objects created using Visual Basic, C++, and JScript reference either Microsoft.VisualBasic.dll, Microsoft.VisualC.dll, or Microsoft.JScript.dll, respectively. These assemblies aren't in the full trust assembly list by default. You must add the appropriate assembly to the full trust list before you add a custom security object. Failure to do so will break the security system, causing all assemblies to fail to load. In this situation, the Caspol.exe -all -reset option won't repair security. To repair security, you must manually edit the security files to remove the custom security object.
	Adds a new code group to the code group hierarchy. You can specify either the <i>parent_label</i> or <i>parent_name</i> . The <i>parent_label</i> argument specifies the label (such as 1. or 1.1.) of the code group that is the parent of the code group being added. The <i>parent_name</i> argument specifies the name of the code group that is the parent of the code group being added. Because <i>parent_label</i> and <i>parent_name</i> can be used interchangeably, Caspol.exe must be able to distinguish between them. Therefore, <i>parent_name</i> can't begin with a number. Additionally, <i>parent_name</i> can only contain A-Z, 0-9 and the underscore character.

Option	Description
<i>parent_name}</i> <i>mship pset_name</i> [<i>flags</i>]	The <i>mship</i> argument specifies the membership condition for the new code group. For more information, see the table of <i>mship</i> arguments later in this section.
	The <i>pset_name</i> argument is the name of the permission set that will be associated with the new code group. You can also set one or more <i>flags</i> for the new group. For more information, see the table of <i>flags</i> arguments later in this section.
-addpset { <i>psfile</i> <i>psfile pset_name</i> } or -ap	Adds a new named permission set to policy. The permission set must be authored in XML and stored in an .xml file. If the XML file contains the name of the permission set, only that file (<i>psfile</i>) is specified. If the XML file doesn't contain the permission set name, you must specify both the XML file name (<i>psfile</i>) and the permission set name (<i>pset_name</i>).
{ <i>named_psfile</i> <i>psfile pset_name</i> }	All permissions used in a permission set must be defined in assemblies contained in the global assembly cache.
-a[ll]	Indicates that all options following this one apply to the machine, user, and enterprise policies. The -all option always refers to the policy of the currently logged-on user. See the -customall option to refer to the user policy of a user other than the current user.
-chgroup { <i>label</i> <i> name</i> } { <i>mship</i> <i>pset_name</i> } or -cg { <i>label name</i> } { <i>mship</i> <i>pset_name</i> } or <i>flags</i> }	Changes a code group's membership condition, permission set, or the settings of the exclusive , levelfinal , name , or description flags. You can specify either the <i>label</i> or <i>name</i> . The <i>label</i> argument specifies the label (such as 1. or 1.1.) of the code group. The <i>name</i> argument specifies the name of the code group to change. Because <i>label</i> and <i>name</i> can be used interchangeably, Caspol.exe must be able to distinguish between them. Therefore, <i>name</i> cannot begin with a number. Additionally, <i>name</i> can only contain A-Z, 0-9 and the underscore character.
-cg { <i>label name</i> } { <i>mship</i> <i>pset_name</i> } or <i>flags</i> }	The <i>pset_name</i> argument specifies the name of the permission set to associate with the code group. See the tables later in this section for information on the <i>mship</i> and <i>flags</i> arguments.
-chgpset <i>psfile</i> <i>pset_name</i> or -cp <i>psfile</i> <i>pset_name</i>	Changes a named permission set. The <i>psfile</i> argument supplies the new definition for the permission set; it's a serialized permission set file in XML format. The <i>pset_name</i> argument specifies the name of the permission set you want to change.
-customall <i>path</i>	Indicates that all options following this one apply to the machine, enterprise, and the specified custom user policies. You must specify the location of the

Option	Description
or -ca <i>path</i>	custom user's security configuration file with the <i>path</i> argument.
-cu [<i>stomuser</i>] <i>path</i>	Allows the administration of a custom user policy that doesn't belong to the user on whose behalf Caspol.exe is currently running. You must specify the location of the custom user's security configuration file with the <i>path</i> argument.
-enterprise or -en	Indicates that all options following this one apply to the enterprise level policy. Users who aren't enterprise administrators don't have sufficient rights to modify the enterprise policy, although they can view it. In non-enterprise scenarios, this policy, by default, doesn't interfere with machine and user policy.
-e[xecution] { on off }	Turns on or off the mechanism that checks for the permission to run before code starts to execute. Note: This switch is removed in .NET Framework 4 and later versions.
-f[orce]	Suppresses the tool's self-destruct test and changes the policy as specified by the user. Normally, Caspol.exe checks whether any policy changes would prevent Caspol.exe itself from running properly; if so, Caspol.exe doesn't save the policy change and prints an error message. To force Caspol.exe to change policy even if this prevents Caspol.exe itself from running, use the -force option.
-h[elp]	Displays command syntax and options for Caspol.exe.
-l[ist]	Lists the code group hierarchy and the permission sets for the specified machine, user, enterprise, or all policy levels. Caspol.exe displays the code group's label first, followed by the name, if it isn't null.
-listdescription or -ld	Lists all code group descriptions for the specified policy level.
-listfulltrust or -lf	Lists the contents of the full trust assembly list for the specified policy level.
-listgroups or -lg	Displays the code groups of the specified policy level or all policy levels. Caspol.exe displays the code group's label first, followed by the name, if it isn't null.

Option	Description
-listpset or -lp	Displays the permission sets for the specified policy level or all policy levels.
-m[achine]	Indicates that all options following this one apply to the machine level policy. Users who aren't administrators don't have sufficient rights to modify the machine policy, although they can view it. For administrators, -machine is the default.
-polchgprompt {on off}	Enables or disables the prompt that is displayed whenever Caspol.exe is run using an option that would cause policy changes.
or	
-pp {on off}	
-quiet	Temporarily disables the prompt that is normally displayed for an option that causes policy changes. The global change prompt setting does not change.
or -q	Use the option only on a single command basis to avoid disabling the prompt for all Caspol.exe commands.
-r[ecover]	Recovers policy from a backup file. Whenever a policy change is made, Caspol.exe stores the old policy in a backup file.
-remfulltrust assembly_file	Removes an assembly from the full trust list of a policy level. This operation should be performed if a permission set that contains a custom permission is no longer used by policy. However, you should remove an assembly that implements a custom permission from the full trust list only if the assembly doesn't implement any other custom permissions that are still being used.
or -rf assembly_file	When you remove an assembly from the list, you should also remove any other assemblies that it depends on.
-remgroup {label name}	Removes the code group specified by either its label or name. If the specified code group has child code groups, Caspol.exe also removes all the child code groups.
or -rg {label name}	
-rempset pset_name	Removes the specified permission set from policy. The <i>pset_name</i> argument indicates which permission set to remove. Caspol.exe removes the permission set only if it isn't associated with any code group. The default (built-in) permission sets can't be removed.
or -rp pset_name	
-reset	Returns policy to its default state and persists it to disk. This is useful whenever a changed policy seems to be beyond repair and you want to start over with the installation defaults. Resetting can also be convenient when you want to use the default policy as a starting point for modifications to specific

Option	Description
-rs	security configuration files. For more information, see Manually Editing the Security Configuration Files .
-resetlockdown or -rsld	Returns policy to a more restrictive version of the default state and persists it to disk; creates a backup of the previous machine policy and persists it to a file called <code>security.config.bac</code> . The locked down policy is similar to the default policy, except that the policy grants no permission to code from the <code>Local Intranet</code> , <code>Trusted Sites</code> , and <code>Internet</code> zones and the corresponding code groups have no child code groups.
-resolvegroup <i>assembly_file</i> or	Shows the code groups that a specific assembly (<i>assembly_file</i>) belongs to. By default, this option displays the machine, user, and enterprise policy levels to which the assembly belongs. To view only one policy level, use this option with either the -machine , -user , or -enterprise option.
-rsg <i>assembly_file</i>	
-resolveperm <i>assembly_file</i> or -rsp <i>assembly_file</i>	Displays all permissions that the specified (or default) level of security policy would grant the assembly if the assembly were allowed to run. The <i>assembly_file</i> argument specifies the assembly. If you specify the -all option, Caspol.exe calculates the permissions for the assembly based on user, machine, and enterprise policy; otherwise, default behavior rules apply.
-s[ecurity] {on off}	Turns code access security on or off. Specifying the -s off option does not disable role-based security. Note: This switch is removed in .NET Framework 4 and later versions. Caution: When code access security is disabled, all code access demands succeed. Disabling code access security makes the system vulnerable to attacks by malicious code such as viruses and worms. Turning off security gains some extra performance but should only be done when other security measures have been taken to help make sure overall system security isn't breached. Examples of other security precautions include disconnecting from public networks, physically securing computers, and so on.
-u[user]	Indicates that all options following this one apply to the user level policy for the user on whose behalf Caspol.exe is running. For nonadministrative users, -user is the default.
-?	Displays command syntax and options for Caspol.exe.

The *mship* argument, which specifies the membership condition for a code group, can be used with the **-addgroup** and **-chggroup** options. Each *mship* argument is implemented as a .NET Framework class. To specify *mship*, use one of the following.

Argument	Description
-allcode	Specifies all code. For more information about this membership condition, see System.Security.Policy.AllMembershipCondition .
-appdir	Specifies the application directory. If you specify -appdir as the membership condition, the URL evidence of code is compared with the application directory evidence of that code. If both evidence values are the same, this membership condition is satisfied. For more information about this membership condition, see System.Security.Policy.ApplicationDirectoryMembershipCondition .
-custom <i>xmlfile</i>	Adds a custom membership condition. The mandatory <i>xmlfile</i> argument specifies the .xml file that contains XML serialization of the custom membership condition.
-hash <i>hashAlg</i> { -hex <i>hashValue</i> -file <i>assembly_file</i> }	Specifies code that has the given assembly hash. To use a hash as a code group membership condition, you must specify either the hash value or the assembly file. For more information about this membership condition, see System.Security.Policy.HashMembershipCondition .
-pub { -cert <i>cert_file_name</i> -file <i>signed_file_name</i> -hex <i>hex_string</i> }	Specifies code that has the given software publisher, as denoted by a certificate file, a signature on a file, or the hexadecimal representation of an X509 certificate. For more information about this membership condition, see System.Security.Policy.PublisherMembershipCondition .
-site <i>website</i>	Specifies code that has the given site of origin. For example:
<code>-site** www.proseware.com</code>	For more information about this membership condition, see System.Security.Policy.SiteMembershipCondition .
-strong - file <i>file_name</i> { <i>name</i> -noname } { <i>version</i> -noversion }	Specifies code that has a specific strong name, as designated by the file name, the assembly name as a string, and the assembly version in the format <i>major.minor.build.revision</i> . For example:
<code>-strong -file myAssembly.exe myAssembly 1.2.3.4</code>	For more information about this membership condition, see System.Security.Policy.StrongNameMembershipCondition .
-url <i>URL</i>	Specifies code that originates from the given URL. The URL must include a protocol, such as <code>http://</code> or <code>ftp://</code> . Additionally, a wildcard character (*) can be used to specify multiple assemblies from a particular URL. Note: Because a URL can be identified using multiple names, using a URL as a membership condition isn't a safe way to ascertain the identity of code. Where possible,

Argument	Description
	<p>use a strong name membership condition, a publisher membership condition, or the hash membership condition.</p> <p>For more information about this membership condition, see System.Security.Policy.UrlMembershipCondition.</p>
-zone <i>zonename</i>	<p>Specifies code with the given zone of origin. The <i>zonename</i> argument can be one of the following values: MyComputer, Intranet, Trusted, Internet, or Untrusted. For more information about this membership condition, see the ZoneMembershipCondition Class.</p>

The *flags* argument, which can be used with the **-addgroup** and **-chggrou**p options, is specified using one of the following.

[+] [Expand table](#)

Argument	Description
-description <i>"description"</i>	<p>If used with the -addgroup option, specifies the description for a code group to add. If used with the -chggroup option, specifies the description for a code group to edit. The <i>description</i> argument must be enclosed in double quotes.</p>
-exclusive {on off}	<p>When set to on, indicates that only the permission set associated with the code group you're adding or modifying is considered when some code fits the membership condition of the code group. When this option is set to off, Caspol.exe considers the permission sets of all matching code groups in the policy level.</p>
-levelfinal {on off}	<p>When set to on, indicates that no policy level below the level in which the added or modified code group occurs is considered. This option is typically used at the machine policy level. For example, if you set this flag for a code group at the machine level and some code matches this code group's membership condition, Caspol.exe does not calculate or apply the user level policy for this code.</p>
-name <i>"name"</i>	<p>If used with the -addgroup option, specifies the scripting name for a code group to add. If used with the -chggroup option, specifies the scripting name for a code group to edit. The <i>name</i> argument must be enclosed in double quotes. The <i>name</i> argument cannot begin with a number, and can only contain A-Z, 0-9, and the underscore character. Code groups can be referred to by this <i>name</i> instead of by their numeric label. The <i>name</i> is also highly useful for scripting purposes.</p>

Remarks

Security policy is expressed using three policy levels: machine policy, user policy, and enterprise policy. The set of permissions that an assembly receives is determined by the

intersection of the permission sets allowed by these three policy levels. Each policy level is represented by a hierarchical structure of code groups. Every code group has a membership condition that determines which code is a member of that group. A named permission set is also associated with each code group. This permission set specifies the permissions the runtime allows code that satisfies the membership condition to have. A code group hierarchy, along with its associated named permission sets, defines and maintains each level of security policy. You can use the **-user**, **-customuser**, **-machine** and **-enterprise** options to set the level of security policy.

For more information about security policy and how the runtime determines which permissions to grant to code, see [Security Policy Management](#).

Referencing Code Groups and Permission Sets

To facilitate references to code groups in a hierarchy, the **-list** option displays an indented list of code groups along with their numerical labels (1, 1.1, 1.1.1, and so on). The other command-line operations that target code groups also use the numerical labels to refer to specific code groups.

Named permission sets are referenced by their names. The **-list** option displays the list of code groups followed by a list of named permission sets available in that policy.

Caspol.exe Behavior

All options except **-s[ecurity] {on | off}** use the version of the .NET Framework that Caspol.exe was installed with. If you run the Caspol.exe that was installed with version *X* of the runtime, the changes apply only to that version. Other side-by-side installations of the runtime, if any, are not affected. If you run Caspol.exe from the command line without being in a directory for a specific runtime version, the tool is executed from the first runtime version directory in the path (usually the most recent runtime version installed).

The **-s[ecurity] {on | off}** option is a computer-wide operation. Turning off code access security terminates security checks for all managed code and for all users on the computer. If side-by-side versions of the .NET Framework are installed, this command turns off security for every version installed on the computer. Although the **-list** option shows that security is turned off, nothing else clearly indicates for other users that security has been turned off.

When a user without administrative rights runs Caspol.exe, all options refer to the user level policy unless the **-machine** option is specified. When an administrator runs

Caspol.exe, all options refer to the machine policy unless the **-user** option is specified.

Caspol.exe must be granted the equivalent of the **Everything** permission set to function. The tool has a protective mechanism that prevents policy from being modified in ways that would prevent Caspol.exe from being granted the permissions it needs to run. If you try to make such changes, Caspol.exe notifies you that the requested policy change will break the tool, and the policy change is rejected. You can turn off this protective mechanism for a given command by using the **-force** option.

Manually Editing the Security Configuration Files

Three security configuration files correspond to the three policy levels supported by Caspol.exe: one for the machine policy, one for a given user's policy, and one for the enterprise policy. These files are created on disk only when machine, user, or enterprise policy is changed using Caspol.exe. You can use the **-reset** option in Caspol.exe to save the default security policy to disk, if needed.

In most cases, manually editing the security configuration files isn't recommended. But there might be scenarios in which modifying these files becomes necessary, such as when an administrator wants to edit the security configuration for a particular user.

Examples

-addfulltrust

Assume that a permission set containing a custom permission has been added to machine policy. This custom permission is implemented in `MyPerm.exe`, and `MyPerm.exe` references classes in `MyOther.exe`. Both assemblies must be added to the full trust assembly list. The following command adds the `MyPerm.exe` assembly to the full trust list for the machine policy.

Console

```
caspol -machine -addfulltrust MyPerm.exe
```

The following command adds the `MyOther.exe` assembly to the full trust list for the machine policy.

Console

```
caspol -machine -addfulltrust MyOther.exe
```

-addgroup

The following command adds a child code group to the root of the machine policy code group hierarchy. The new code group is a member of the **Internet** zone and is associated with the **Execution** permission set.

Console

```
caspol -machine -addgroup 1. -zone Internet Execution
```

The following command adds a child code group that gives the share `\netserver\netshare` local intranet permissions.

Console

```
caspol -machine -addgroup 1. -url \\netserver\netshare\* LocalIntranet
```

-addpset

The following command adds the `Mypset` permission set to the user policy.

Console

```
caspol -user -addpset Mypset.xml Mypset
```

-chggrouop

The following command changes the permission set in the user policy of the code group labeled 1.2. to the **Execution** permission set.

Console

```
caspol -user -chggrouop 1.2. Execution
```

The following command changes the membership condition in the default policy of the code group labeled 1.2.1. and changes the setting of the **exclusive** flag. The membership condition is defined to be code that originates from the **Internet** zone and the **exclusive** flag is switched on.

Console

```
caspol -chggroup 1.2.1. -zone Internet -exclusive on
```

-chgpset

The following command changes the permission set with name `Mypset` to the permission set contained in `newpset.xml`. Note that the current release doesn't support changing permission sets that are being used by the code group hierarchy.

Console

```
caspol -chgpset Mypset newpset.xml
```

-force

The following command causes the user policy's root code group (labeled 1) to be associated with the **Nothing** named permission set. This prevents Caspol.exe from running.

Console

```
caspol -force -user -chggroup 1 Nothing
```

-recover

The following command recovers the most recently saved machine policy.

Console

```
caspol -machine -recover
```

-remgroup

The following command removes the code group labeled 1.1. If this code group has any child code groups, those groups are also deleted.

Console

```
caspol -remgroup 1.1.
```

-rempset

The following command removes the **Execution** permission set from the user policy.

Console

```
caspol -user -rmpset Execution
```

The following command removes `MyPset` from the user policy level.

Console

```
caspol -rmpset MyPset
```

-resolvegroup

The following command shows all code groups of the machine policy that `myassembly` belongs to.

Console

```
caspol -machine -resolvegroup myassembly
```

The following command shows all code groups of the machine, enterprise, and specified custom user policy that `myassembly` belongs to.

Console

```
caspol -customall "c:\config_test\security.config" -resolvegroup myassembly
```

-resolveperm

The following command calculates the permissions for `testassembly` based on the machine and user policy levels.

Console

```
caspol -all -resolveperm testassembly
```

See also

- [Tools](#)
- [Developer command-line shells](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Cert2spc.exe (Software Publisher Certificate Test Tool)

Article • 09/15/2021

The Software Publisher Certificate Test tool creates a Software Publisher's Certificate (SPC) from one or more X.509 certificates. Cert2spc.exe is for test purposes only. You can obtain a valid SPC from a Certification Authority such as VeriSign or Thawte. For more information about creating X.509 certificates, see [Makecert.exe \(Certificate Creation Tool\)](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
cert2spc cert1.cer | crl1.crl [... certN.cer | crlN.crl] outputSPCfile.spc
```

Parameters

Argument	Description
<code>certN.cer</code>	The name of an X.509 certificate to include in the SPC file. You can specify multiple names separated by spaces.
<code>crlN.crl</code>	The name of a certificate revocation list to include in the SPC file. You can specify multiple names separated by spaces.
<code>outputSPCfile.spc</code>	The name of the PKCS #7 object that will contain the X.509 certificates.

Option	Description
<code>/?</code>	Displays command syntax and options for the tool.

Examples

The following command creates an SPC from `myCertificate.cer` and places it in `mySPCFile.spc`.

Console

```
cert2spc myCertificate.cer mySPCFile.spc
```

The following command creates an SPC from `oneCertificate.cer` and `twoCertificate.cer`, and places it in `mySPCFile.spc`.

Console

```
cert2spc oneCertificate.cer twoCertificate.cer mySPCFile.spc
```

See also

- [Tools](#)
- [Makecert.exe \(Certificate Creation Tool\)](#)
- [Developer command-line shells](#)

Certmgr.exe (Certificate Manager Tool)

Article • 05/22/2023

The Certificate Manager tool (Certmgr.exe) manages certificates, certificate trust lists (CTLs), and certificate revocation lists (CRLs).

The Certificate Manager is installed with the Windows 10 SDK. To start the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

ⓘ Note

The Certificate Manager tool (Certmgr.exe) is a command-line utility, whereas Certificates (Certmgr.msc) is a Microsoft Management Console (MMC) snap-in. Because Certmgr.msc is usually found in the Windows System directory, entering `certmgr` at the command line may load the Certificates MMC snap-in even if you've opened the Developer Command Prompt for Visual Studio. This occurs because the path to the snap-in precedes the path to the Certificate Manager tool in the `Path` environment variable. If you encounter this problem, you can execute Certmgr.exe commands by specifying the path to the executable, for example, `%ProgramFiles(x86)%\Windows Kits\10\bin\10.0.22000.0\arm64\certmgr.exe`.

For an overview of X.509 certificates, see [Working with Certificates](#).

Syntax

At the command prompt, enter the following:

Console

```
certmgr [/add | /del | /put] [options]
[/s[ /r registryLocation]] [sourceStorename]
[/s[ /r registryLocation]] [destinationStorename]
```

Parameters

Argument	Description
<code>sourceStorename</code>	The certificate store that contains the existing certificates, CTLs, or CRLs to add, delete, save, or display. This can be a store file or a systems store.

Argument	Description
<i>destinationStorename</i>	The output certificate store or file.
Option	Description
/add	Adds certificates, CTLs, and CRLs to a certificate store.
/all	Adds all entries when used with /add. Deletes all entries when used with /del. Displays all entries when used without the /add or /del options. The /all option cannot be used with /put.
/c	Adds certificates when used with /add. Deletes certificates when used with /del. Saves certificates when used with /put. Displays certificates when used without the /add, /del, or /put option.
/CRL	Adds CRLs when used with /add. Deletes CRLs when used with /del. Saves CRLs when used with /put. Displays CRLs when used without the /add, /del, or /put option.
/CTL	Adds CTLs when used with /add. Deletes CTLs when used with /del. Saves CTLs when used with /put. Displays CTLs when used without the /add, /del, or /put option.
/del	Deletes certificates, CTLs, and CRLs from a certificate store.
/e <i>encodingType</i>	Specifies the certificate encoding type. The default is <code>x509 ASN_ENCODING</code> .
/f <i>dwFlags</i>	Specifies the store open flag. This is the <i>dwFlags</i> parameter passed to <code>CertOpenStore</code> . The default value is <code>CERT_SYSTEM_STORE_CURRENT_USER</code> . This option is considered only if the /y option is used.
/h[elp]	Displays command syntax and options for the tool.
/n <i>nam</i>	Specifies the common name of the certificate to add, delete, or save. This option can only be used with certificates; it cannot be used with CTLs or CRLs.
/put	Saves an X.509 certificate, CTL, or CRL from a certificate store to a file. The file is saved in X.509 format. You can use the /7 option with the /put option to save the file in PKCS #7 format. The /put option must be followed by either /c, /CTL, or /CRL. The /all option cannot be used with /put.
/r <i>location</i>	Identifies the registry location of the system store. This option is considered only if you specify the /s option. <i>location</i> must be one of the following: <ul style="list-style-type: none"> - <code>currentUser</code> indicates that the certificate store is under the <code>HKEY_CURRENT_USER</code> key. This is the default. - <code>localMachine</code> indicates that the certificate store is under the <code>HKEY_LOCAL_MACHINE</code> key.

Option	Description
/s	Indicates that the certificate store is a system store. If you do not specify this option, the store is considered to be a StoreFile .
/sha1 <i>sha1Hash</i>	Specifies the SHA1 hash of the certificate, CTL, or CRL to add, delete, or save.
/v	Specifies verbose mode; displays detailed information about certificates, CTLs, and CRLs. This option cannot be used with the /add, /del, or /put options.
/y provider	Specifies the store provider name.
/7	Saves the destination store as a PKCS #7 object.
/?	Displays command syntax and options for the tool.

Remarks

Certmgr.exe performs the following basic functions:

- Displays certificates, CTLs, and CRLs to the console.
- Adds certificates, CTLs, and CRLs to a certificate store.
- Deletes certificates, CTLs, and CRLs from a certificate store.
- Saves an X.509 certificate, CTL, or CRL from a certificate store to a file.

Certmgr.exe works with two types of certificate stores: **StoreFile** and system store. It's not necessary to specify the type of certificate store; Certmgr.exe can identify the store type and perform the appropriate operations.

Running Certmgr.exe without specifying any options launches the certmgr.msc snap-in, which has a GUI that helps with the certificate management tasks that are also available from the command line. The GUI provides an import wizard, which copies certificates, CTLs, and CRLs from your disk to a certificate store.

You can find the names of X509Certificate stores for the `sourceStorename` and `destinationStorename` parameters by compiling and running the following code.

```
C#
using System;
using System.Security.Cryptography.X509Certificates;

public class Example
{
    public static void Main()
    {
```

```
        foreach (var storeValue in Enum.GetValues(typeof(StoreName))) {
            X509Store store = new X509Store((StoreName) storeValue);
            store.Open(OpenFlags.ReadOnly);
            Console.WriteLine(store.Name);
        }
    }
}
```

For more information about certificates, see [Working with Certificates](#).

Examples

The following command displays a default system store called `my` with verbose output.

Console

```
certmgr /v /s my
```

The following command adds all the certificates in a file called `myFile.ext` to a new file called `newFile.ext`.

Console

```
certmgr /add /all /c myFile.ext newFile.ext
```

The following command adds the certificate in a file named `testcert.cer` to the `my` system store.

Console

```
certmgr /add /c testcert.cer /s my
```

The following command adds the certificate in a file named `TrustedCert.cer` to the root certificate store.

Console

```
certmgr /c /add TrustedCert.cer /s root
```

The following command saves a certificate with the common name `myCert` in the `my` system store to a file called `newCert.cer`.

Console

```
certmgr /add /c /n myCert /s my newCert.cer
```

The following command deletes all CTLs in the `my` system store and saves the resulting store to a file called `newStore.str`.

Console

```
certmgr /del /all /ctl /s my newStore.str
```

The following command saves a certificate in the `my` system store in the file `newFile`. You will be prompted to enter the certificate number from `my` to put in `newFile`.

Console

```
certmgr /put /c /s my newFile
```

See also

- [Tools](#)
- [CertMgr \(Windows app development\)](#)
- [Makecert.exe \(Certificate Creation Tool\)](#)
- [Developer command-line shells](#)

Clrver.exe (CLR Version Tool)

Article • 09/15/2021

The CLR Version tool (Clrver.exe) reports all the installed versions of the common language runtime (CLR) on the computer.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, enter the following command:

Syntax

Console

```
clrver [option]
```

Options

Option	Description
-all	Displays all processes on the computer that are using the CLR.
pid	Displays the version(s) of the CLR used by the process that has the specified process ID (PID).
-?	Displays command syntax and options for the tool.

Remarks

If you call Clrver.exe with no options, it displays all installed CLR versions. If you specify a PID for another user, you must have administrative permissions to obtain the version information.

Note

In Windows Vista and later, User Account Control (UAC) determines the privileges of a user. If you are a member of the Built-in Administrators group, you are assigned two run-time access tokens: a standard user access token and an administrator access token. By default, you are in the standard user role. To execute

code that requires administrative permission, you must first elevate your privileges from standard user to administrator. You can do this when you start the command prompt by right-clicking the command prompt icon and indicating that you want to run as an administrator.

Attempting to determine the CLR version for SYSTEM, LOCAL SERVICE, and NETWORK SERVICE processes results in a message indicating that the PID doesn't exist.

Examples

The following command displays all the versions of the CLR installed on the computer.

```
clrver
```

The following command displays the version of the CLR used by process 128.

```
clrver 128
```

The following command displays all the managed processes and the version of the CLR they are using.

```
Clrver -all
```

See also

- [Tools](#)
- [Developer command-line shells](#)

CorFlags.exe (CorFlags Conversion Tool)

Article • 05/18/2022

The CorFlags Conversion tool allows you to configure the CorFlags section of the header of a portable executable image.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
CorFlags.exe assembly [options]
```

Parameters

Required parameter	Description
<code>assembly</code>	The name of the assembly for which to configure the CorFlags.

Option	Description
<code>-32BIT[REQ]+</code>	Sets the 32BITREQUIRED flag.
<code>-32BIT[REQ]-</code>	Clears the 32BITREQUIRED flag.
<code>-32BITPREF+</code>	Sets the 32BITPREFERRED flag. The app runs as a 32-bit process even on 64-bit platforms. Set this flag only on EXE files. If the flag is set on a DLL, the DLL fails to load in 64-bit processes, and a BadImageFormatException exception is thrown. An EXE file with this flag can be loaded into a 64-bit process. New in the .NET Framework 4.5.
<code>-32BITPREF-</code>	Clears the 32BITPREFERRED flag. New in the .NET Framework 4.5.
<code>-?</code>	Displays command syntax and options for the tool.

Option	Description
-Force	Forces an update even if the assembly is strong-named. Important: If you update a strong-named assembly, you must sign it again before executing its code.
-help	Displays command syntax and options for the tool.
-ILONLY+	Sets the ILONLY flag.
-ILONLY-	Clears the ILONLY flag.
-nologo	Suppresses the Microsoft startup banner display.
-	Reverts the CLR header version to 2.0.
RevertCLRHeader	
-	Upgrades the CLR header version to 2.5. Note: Assemblies must have a CLR header version of 2.5 or greater to run natively.
UpgradeCLRHeader	

Remarks

If no options are specified, the CorFlags Conversion tool displays the flags for the specified assembly.

For more information, see section II.25.3.3.1 Runtime flags of the [ECMA-335 specification](#). For information about the PE flag, see [Optional Header \(Image Only\)](#).

See also

- [Tools](#)
- [64-bit Applications](#)
- [Developer command-line shells](#)

Fuslogvw.exe (Assembly Binding Log Viewer)

Article • 03/30/2023

The Assembly Binding Log Viewer displays details for assembly binds. This information helps you diagnose why the .NET Framework cannot locate an assembly at run time. These failures are usually the result of an assembly deployed to the wrong location, a native image that is no longer valid, or a mismatch in version numbers or cultures. The common language runtime's failure to locate an assembly typically shows up as a [TypeLoadException](#) in your application.

ⓘ Important

You must run fuslogvw.exe with administrator privileges.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#) with administrator credentials.

At the command prompt, enter the following command:

```
Console
```

```
fuslogvw
```

The viewer displays an entry for each failed assembly bind. For each failure, the viewer describes:

- the application that initiated the bind
- the assembly the bind is for, including name, version, culture and public key
- the date and time of the failure

How to...

- [Change the log location view](#)
- [View details about a specific failure](#)
- [Delete entries](#)
- [Refresh the user interface](#)
- [Change the log settings](#)

- [View the About dialog](#)

Change the log location view

1. Select the **Default** option button to view bind failures for all application types. By default, log entries are stored in per-user directories on disk in the wininet cache.
2. Select the **Custom** option button to view bind failures in a custom directory that you specify. You must specify the custom location where you want the runtime to store the logs by setting the custom log location in the **Log Settings** dialog to a valid directory name. This directory should be clean, and only contain files that the runtime generates. If it contains an executable that generates a failure to be logged, the failure will not be logged because the tool tries to create a directory with the same name as the executable. In addition, an attempt to run an executable from the log location will fail.

Note

The default bind location is preferable to the custom bind location. The runtime stores the default bind location in the wininet cache, and therefore automatically cleans it out. If you specify a custom bind location, you are responsible for cleaning it out.

View details about a specific failure

1. Select the application name of the desired entry in the viewer.
2. Click the **View Log** button. Alternately, you can double-click the selected entry.

The tool displays the following details about the selected bind failure:

- The specific reason the bind failed, such as "file not found" or "version mismatch".
- Information about the application that initiated the bind, including its name, the application's root directory (AppBase), and a description of the private search path, if there is one.
- The identity of the assembly the tool is looking for.
- A description of any Application, Publisher, or Administrator version policies that have been applied.

- Whether the assembly was found in the global assembly cache.
- A list of all probing URLs.

The following sample log entry shows detailed information about a failed assembly bind.

Output

```
*** Assembly Binder Log Entry (3/5/2007 @ 12:54:20 PM) ***

The operation failed.
Bind result: hr = 0x80070002. The system cannot find the file specified.

Assembly manager loaded from:
C:\WINNT\Microsoft.NET\Framework\v2.0.50727\fusion.dll
Running under executable C:\Program
Files\Microsoft.NET\FrameworkSDK\Samples\Tutorials\resourcesandlocalization\
graphic\cs\graphicfailtest.exe
--- A detailed error log follows.

==== Pre-bind state information ====
LOG: DisplayName = graphicfailtest.resources, Version=0.0.0.0, Culture=en-
US, PublicKeyToken=null
(Fully-specified)
LOG: Appbase = C:\Program
Files\Microsoft.NET\FrameworkSDK\Samples\Tutorials\resourcesandlocalization\
graphic\cs\
LOG: Initial PrivatePath = NULL
LOG: Dynamic Base = NULL
LOG: Cache Base = NULL
LOG: AppName = NULL
Calling assembly : graphicfailtest, Version=0.0.0.0, Culture=neutral,
PublicKeyToken=null.
====

LOG: Processing DEVPATH.
LOG: DEVPATH is not set. Falling through to regular bind.
LOG: Policy not being applied to reference at this time (private, custom,
partial, or location-based assembly bind).
LOG: Post-policy reference: graphicfailtest.resources, Version=0.0.0.0,
Culture=en-US, PublicKeyToken=null
LOG: Attempting download of new URL file:///C:/Program
Files/Microsoft.NET/FrameworkSDK/Samples/Tutorials/resourcesandlocalization/
graphic/cs/graphicfailtest.resources.DLL.
LOG: Attempting download of new URL file:///C:/Program
Files/Microsoft.NET/FrameworkSDK/Samples/Tutorials/resourcesandlocalization/
graphic/cs/graphicfailtest.resources/graphicfailtest.resources.DLL.
LOG: Attempting download of new URL file:///C:/Program
Files/Microsoft.NET/FrameworkSDK/Samples/Tutorials/resourcesandlocalization/
graphic/cs/graphicfailtest.resources.EXE.
LOG: Attempting download of new URL file:///C:/Program
Files/Microsoft.NET/FrameworkSDK/Samples/Tutorials/resourcesandlocalization/
```

```
graphic/cs/graphicfailtest.resources/graphicfailtest.resources.EXE.  
LOG: All probing URLs attempted and failed.
```

Delete entries

To delete a single entry from the log:

1. Select an entry in the viewer.
2. Click the **Delete Entry** button.

To delete all entries from the log:

- Click the **Delete All** button.

Refresh the user interface

- Click the **Refresh** button. The viewer does not automatically detect new log entries while it is running. You must use the **Refresh** button to display them.

Change the log settings

Click the **Settings** button to open the **Log Settings** dialog.

View the About dialog

Click the **About** button.

Binding logs for native images

By default, Fuslogvw.exe logs normal assembly bind requests. Alternatively, you can log assembly binds for native images that were created using the [Ngen.exe \(Native Image Generator\)](#).

Log assembly binds for native images

- In the **Log Categories** group, select the **Native Images** option button.

The following log shows a failure caused by a dependency that did not exist when the native image was created for the application. If the dependencies at run time differ from the dependencies when Ngen.exe is run, binding to a native image is not allowed.

Output

```
*** Assembly Binder Log Entry (12/8/2006 @ 5:22:07 PM) ***

The operation failed.
Bind result: hr = 0x80070002. The system cannot find the file specified.

Assembly manager loaded from:
E:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorwks.dll
Running under executable E:\test\App.exe
--- A detailed error log follows.

LOG: Start binding of native image App, Version=0.0.0.0, Culture=neutral,
PublicKeyToken=null.
LOG: IL assembly loaded from E:\test\App.exe.
LOG: Start validating native image App, Version=0.0.0.0, Culture=neutral,
PublicKeyToken=null.
LOG: Start validating all the dependencies.
LOG: [Level 1]Start validating native image dependency mscorelib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089.
LOG: Dependency evaluation succeeded.
LOG: [Level 1]Start validating IL dependency b, Version=0.0.0.0,
Culture=neutral, PublicKeyToken=null.
WRN: Dependency assembly was not found at ngen time, but is found at binding
time. Disallow using this native image.
WRN: No matching native image found.
LOG: Bind to native image assembly did not succeed. Use IL image.
```

The following log shows a native image binding failure that occurred because the security settings on the computer when the application was run were different from the security settings at the time the native image was created.

Output

```
*** Assembly Binder Log Entry (12/8/2006 @ 5:29:09 PM) ***

The operation failed.
Bind result: hr = 0x80004005. Unspecified error

Assembly manager loaded from:
E:\Windows\Microsoft.NET\Framework64\v2.0.50727\mscorwks.dll
Running under executable E:\test\Application101622.exe
--- A detailed error log follows.

LOG: Start binding of native image Application101622, Version=0.0.0.0,
Culture=neutral, PublicKeyToken=null.
LOG: IL assembly loaded from E:\test\Application101622.exe.
LOG: Start validating native image Application101622, Version=0.0.0.0,
Culture=neutral, PublicKeyToken=null.
LOG: Start validating all the dependencies.
LOG: [Level 1]Start validating native image dependency mscorelib,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089.
```

```
LOG: Dependency evaluation succeeded.  
LOG: [Level 1]Start validating IL dependency Dependency101622,  
Version=0.0.0.0, Culture=neutral, PublicKeyToken=null.  
LOG: Dependency evaluation succeeded.  
LOG: Validation of dependencies succeeded.  
LOG: Start loading all the dependencies into load context.  
LOG: Loading of dependencies succeeded.  
LOG: Bind to native image succeeded.  
Native image has correct version information.  
Attempting to use native image  
E:\Windows\assembly\NativeImages_v2.0.50727_64\Application101622\1ac7fadabec  
4f72575d807501e9fdc72\Application101622.ni.exe.  
Rejecting native image because it failed the security check. The assembly's  
permissions must have changed since the time it was generated, or it is  
running with a different security context.  
Discarding native image.
```

The Log Settings dialog

You can use the **Log Settings** dialog to perform the following actions.

To disable logging

- Select the **Log disabled** option button. Note that this option is selected by default.

To log assembly binds in exceptions

- Select the **Log in exception text** option button. Only the least detailed fusion log information is logged in exception text. To view full information, use one of the other settings.

See the Important note regarding assemblies that are loaded as domain neutral.

To log assembly bind failures

- Select the **Log bind failures to disk** option button.

See the Important note regarding assemblies that are loaded as domain neutral.

To log all assembly binds

- Select the **Log all binds to disk** option button.

See the Important note regarding assemblies that are loaded as domain neutral.

ⓘ Important

When an assembly is loaded as domain neutral, for example by setting the [LoaderOptimization](#) property to [LoaderOptimization.MultiDomain](#) or [LoaderOptimization.MultiDomainHost](#), turning on logging might leak memory in some cases. This can happen if a log entry is made when a domain-neutral module is loaded into an application domain, and later the application domain is unloaded. The log entry might not be released until the process ends. Some debuggers automatically turn on logging.

To enable a custom log path

1. Select the **Enable custom log path** option button.
2. Enter the path into the **Custom log path** text box.

ⓘ Note

The [Assembly Binding Log Viewer \(Fuslogvw.exe\)](#) uses the internet file cache to store its binding log. Due to occasional corruption in the cache, the [Assembly Binding Log Viewer \(Fuslogvw.exe\)](#) can sometimes stop showing new binding logs in the viewing window. As a result of this corruption, the .NET binding infrastructure (fusion) cannot write to or read from the binding log. (This issue is not encountered if you use a custom log path.) To fix the corruption and allow fusion to show binding logs again, clear the internet file cache by deleting temporary internet files from the **Browsing history** section under Internet properties.

If your unmanaged application hosts the common language runtime by implementing the [IHostAssemblyManager](#) and [IHostAssemblyStore](#) interfaces, log entries can't be stored in the wininet cache. To view log entries for custom hosts that implement these interfaces, you must specify an alternate log path.

To enable logging for apps running in the Windows app container

1. Enable a custom log path, as described in the previous procedure. By default, apps that are running in the Windows app container have limited access to the hard disk. The directory you specify will have read/write access for all apps in the app container.

2. Select the **Enable immersive logging** check box.

 **Note**

This box is enabled only on Windows 8 or later.

See also

- [TypeLoadException](#)
- [Tools](#)
- [Global Assembly Cache](#)
- [How the Runtime Locates Assemblies](#)
- [Developer command-line shells](#)

Gacutil.exe (Global Assembly Cache tool)

Article • 07/23/2022

The Global Assembly Cache tool allows you to view and manipulate the contents of the global assembly cache and download cache.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
gacutil [options] [assemblyName | assemblyPath | assemblyListFile]
```

Parameters

Argument	Description
<i>assemblyName</i>	The name of an assembly. You can supply either a partially specified assembly name such as <code>myAssembly</code> or a fully specified assembly name such as <code>myAssembly, Version=2.0.0.0, Culture=neutral,</code> <code>PublicKeyToken=0038abc9deabfle5.</code>
<i>assemblyPath</i>	The name of a file that contains an assembly manifest.
<i>assemblyListFile</i>	The path to an ANSI text file that lists assemblies to install or uninstall. To use a text file to install assemblies, specify the path to each assembly on a separate line in the file. The tool interprets relative paths, relative to the location of the <i>assemblyListFile</i> . To use a text file to uninstall assemblies, specify the fully qualified assembly name for each assembly on a separate line in the file. See the <i>assemblyListFile</i> contents examples later in this topic.

Option	Description
<code>/cdl</code>	Deletes the contents of the download cache.

Option	Description
/f	Specify this option with the /i or /il options to force an assembly to reinstall. If an assembly with the same name already exists in the global assembly cache, the tool overwrites it.
/h[elp]	Displays command syntax and options for the tool.
/i <i>assemblyPath</i>	Installs an assembly into the global assembly cache.
/if <i>assemblyPath</i>	Installs an assembly into the global assembly cache. If an assembly with the same name already exists in the global assembly cache, the tool overwrites it. Specifying this option is equivalent to specifying the /i and /f options together.
/il <i>assemblyListFile</i>	Installs one or more assemblies specified in <i>assemblyListFile</i> into the global assembly cache.
<i>ir</i> <i>assemblyPath</i> <i>scheme</i> <i>id</i> <i>description</i>	Installs an assembly into the global assembly cache and adds a reference to count the assembly. You must specify the <i>assemblyPath</i> , <i>scheme</i> , <i>id</i> , and <i>description</i> parameters with this option. For a description of the valid values you can specify for these parameters, see the /r option. Specifying this option is equivalent to specifying the /i and /r options together.
/l [<i>assemblyName</i>]	Lists the contents of the global assembly cache. If you specify the <i>assemblyName</i> parameter, the tool lists only the assemblies matching that name.
/ldl	Lists the contents of the downloaded files cache.
/lr [<i>assemblyName</i>]	Lists all assemblies and their corresponding reference counts. If you specify the <i>assemblyName</i> parameter, the tool lists only the assemblies matching that name and their corresponding reference counts.
/nologo	Suppresses the Microsoft startup banner display.

Option	Description
/r [assemblyName assemblyPath]	Specifies a traced reference to an assembly or assemblies to install or uninstall. Specify this option with the /i, /il, /u, or /ul options.
scheme id	To install an assembly, specify the <i>assemblyPath</i> , <i>scheme</i> , <i>id</i> , and <i>description</i> parameters with this option. To uninstall an assembly, specify the <i>assemblyName</i> , <i>scheme</i> , <i>id</i> , and <i>description</i> parameters.
description	To remove a reference to an assembly, you must specify the same <i>scheme</i> , <i>id</i> , and <i>description</i> parameters that were specified with the /i and /r (or /ir) options when the assembly was installed. If you are uninstalling an assembly, the tool also removes the assembly from the global assembly cache if it is the last reference to remove and if Windows Installer has no outstanding references to the assembly.
	The <i>scheme</i> parameter specifies the type of installation scheme. You can specify one of the following values:
	<ul style="list-style-type: none"> - UNINSTALL_KEY: Specify this value if the installer adds the application to Add/Remove Programs in Microsoft Windows. Applications add themselves to Add/Remove Programs by adding a registry key to HKLM\Software\Microsoft\Windows\CurrentVersion. - FILEPATH: Specify this value if the installer does not add the application to Add/Remove Programs. - OPAQUE: Specify this value if supplying a registry key or file path does not apply to your installation scenario. This value allows you to specify custom information for the <i>id</i> parameter.
	The value to specify for the <i>id</i> parameter depends on the value specified for the <i>scheme</i> parameter:
	<ul style="list-style-type: none"> - If you specify UNINSTALL_KEY for the <i>scheme</i> parameter, specify the name of the application set in the HKLM\Software\Microsoft\Windows\CurrentVersion registry key. For example, if the registry key is HKLM\Software\Microsoft\Windows\CurrentVersion\MyApp, specify MyApp for the <i>id</i> parameter. - If you specify FILEPATH for the <i>scheme</i> parameter, specify the full path to the executable file that installs the assembly as the <i>id</i> parameter. - If you specify OPAQUE for the <i>scheme</i> parameter, you can supply any piece of data as the <i>id</i> parameter. The data you specify must be enclosed in quotation marks ("").
	The <i>description</i> parameter allows you to specify descriptive text about the application to install. This information is displayed when references are enumerated.
/silent	Suppresses the display of all output.

Option	Description
/u <i>assemblyName</i>	Uninstalls an assembly from the global assembly cache.
/uf <i>assemblyName</i>	Forces a specified assembly to uninstall by removing all references to the assembly. Specifying this option is equivalent to specifying the /u and /f options together. Note: You cannot use this option to remove an assembly that was installed using Microsoft Windows Installer. If you attempt this operation, the tool displays an error message.
/ul <i>assemblyListFile</i>	Uninstalls one or more assemblies specified in <i>assemblyListFile</i> from the global assembly cache.
/u[ngen] <i>assemblyName</i>	Uninstalls a specified assembly from the global assembly cache. If the specified assembly has existing reference counts, the tool displays the reference counts and does not remove the assembly from the global assembly cache. Note: In .NET Framework version 2.0, <code>/ungen</code> is not supported. Instead, use the <code>uninstall</code> command of the Ngen.exe (Native Image Generator) . In the .NET Framework versions 1.0 and 1.1, specifying <code>/ungen</code> causes Gacutil.exe to remove the assembly from the native image cache. This cache stores the native images for assemblies that have been created using the Ngen.exe (Native Image Generator) .
/ur <i>assemblyName</i> <i>scheme</i> <i>id</i> <i>description</i>	Uninstalls a reference to a specified assembly from the global assembly cache. To remove a reference to an assembly, you must specify the same <i>scheme</i> , <i>id</i> , and <i>description</i> parameters that were specified with the /i and /r (or /ir) options when the assembly was installed. For a description of the valid values you can specify for these parameters, see the /r option. Specifying this option is equivalent to specifying the /u and /r options together.
/?	Displays command syntax and options for the tool.

Remarks

ⓘ Note

You must have administrator privileges to use Gacutil.exe.

Specifically, Gacutil.exe allows you to install assemblies into the cache, remove them from the cache, and list the contents of the cache.

Gacutil.exe provides options that support reference counting similar to the reference counting scheme supported by Windows Installer. You can use Gacutil.exe to install two applications that install the same assembly; the tool keeps track of the number of references to the assembly. As a result, the assembly will remain on the computer until both applications are uninstalled. If you are using Gacutil.exe for actual product installations, use the options that support reference counting. Use the /i and /r options together to install an assembly and add a reference to count it. Use the /u and /r options together to remove a reference count for an assembly. Be aware that using the /i and /u options alone does not support reference counting. These options are appropriate for use during product development but not for actual product installations.

Use the /il or /ul options to install or uninstall a list of assemblies stored in an ANSI text file. The contents of the text file must be formatted correctly. To use a text file to install assemblies, specify the path to each assembly on a separate line in the file. The following example demonstrates the contents of a file containing assemblies to install.

```
text  
  
myAssembly1.dll  
myAssembly2.dll  
myAssembly3.dll
```

To use a text file to uninstall assemblies, specify the fully qualified assembly name for each assembly on a separate line in the file. The following example demonstrates the contents of a file containing assemblies to uninstall.

```
text  
  
myAssembly1,Version=1.1.0.0,Culture=en,PublicKeyToken=874e23ab874e23ab  
myAssembly2,Version=1.1.0.0,Culture=en,PublicKeyToken=874e23ab874e23ab  
myAssembly3,Version=1.1.0.0,Culture=en,PublicKeyToken=874e23ab874e23ab
```

ⓘ Note

Attempting to install an assembly with a filename longer than between 79 and 91 characters (excluding the file extension) can result in the following error:

```
Output
```

```
Failure adding assembly to the cache: The file name is too long.
```

This is because internally Gacutil.exe constructs a path of up to MAX_PATH characters that consists of the following elements:

- GAC Root - 34 chars (ie. `C:\Windows\Microsoft.NET\assembly\`)
- Architecture - 7 or 9 chars (ie. `GAC_32\`, `GAC_64\`, `GAC_MSIL\`)
- AssemblyName - Up to 91 chars, depending on the size of the other elements (eg. `System.Xml.Linq\`)
- AssemblyInfo - 31 to 48 chars or more consisting of:
 - Framework - 5 chars (eg. `v4.0_\`)
 - AssemblyVersion - 8 to 24 chars (eg. `9.0.1000.0_\`)
 - AssemblyLanguage - 1 to 8 chars (eg. `de_\`, `sr-Cyr1_\`)
 - PublicKey - 17 chars (eg. `31bf3856ad364e35\`)
- DllFileName - Up to 91 + 4 chars (ie. `<AssemblyName>.dll`)

Examples

The following command installs the assembly `mydll.dll` into the global assembly cache.

Console

```
gacutil /i mydll.dll
```

The following command removes the assembly `hello` from the global assembly cache as long as no reference counts exist for the assembly.

Console

```
gacutil /u hello
```

Note that the previous command might remove more than one assembly from the assembly cache because the assembly name is not fully specified. For example, if both version 1.0.0.0 and 3.2.2.1 of `hello` are installed in the cache, the command `gacutil /u hello` removes both of the assemblies.

Use the following example to avoid removing more than one assembly. This command removes only the `hello` assembly that matches the fully specified version number, culture, and public key.

Console

```
gacutil /u hello, Version=1.0.0.1,
Culture="de", PublicKeyToken=45e343aae32233ca
```

The following command installs the assemblies specified in the file `assemblyList.txt` into the global assembly cache.

```
Console  
gacutil /il assemblyList.txt
```

The following command removes the assemblies specified in the file `assemblyList.txt` from the global assembly cache.

```
Console  
gacutil /ul assemblyList.txt
```

The following command installs `myD11.dll` into the global assembly cache and adds a reference to count it. The assembly `myD11.dll` is used by the application `MyApp`. The `UNINSTALL_KEY MyApp` parameter specifies the registry key that adds `MyApp` to Add/Remove Programs in Windows. The description parameter is specified as `My Application Description`.

```
Console  
gacutil /i /r myD11.dll UNINSTALL_KEY MyApp "My Application Description"
```

The following command installs `myD11.dll` into the global assembly cache and adds a reference to count it. The scheme parameter, `FILEPATH`, and the id parameter, `c:\applications\myApp\myApp.exe`, specify the path to the application that is installing `myD11.dll`. The description parameter is specified as `MyApp`.

```
Console  
gacutil /i /r myD11.dll FILEPATH c:\applications\myApp\myApp.exe MyApp
```

The following command installs `myD11.dll` into the global assembly cache and adds a reference to count it. The scheme parameter, `OPAQUE`, allows you to customize the id and description parameters.

```
Console  
gacutil /i /r myD11.dll OPAQUE "Insert custom application details here"  
"Insert Custom description information here"
```

The following command removes the reference to `myDll.dll` by the application `myApp`. If this is the last reference to the assembly, it will also remove the assembly from the global assembly cache.

Console

```
gacutil /u /r myDll.dll FILEPATH c:\applications\myApp\myApp.exe MyApp
```

The following command lists the contents of the global assembly cache.

Console

```
gacutil /l
```

See also

- [Tools](#)
- [Global Assembly Cache](#)
- [Regasm.exe \(Assembly Registration Tool\)](#)
- [Developer command-line shells](#)

Ilasm.exe (IL Assembler)

Article • 07/23/2022

The IL Assembler generates a portable executable (PE) file from intermediate language (IL) assembly. (For more information on IL, see [Managed Execution Process](#).) You can run the resulting executable, which contains IL and the required metadata, to determine whether the IL performs as expected.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
ilasm [options] filename [[options]filename...]
```

Parameters

Argument	Description
<code>filename</code>	The name of the .il source file. This file consists of metadata declaration directives and symbolic IL instructions. Multiple source file arguments can be supplied to produce a single PE file with <i>Ilasm.exe</i> . Note: Ensure that the last line of code in the .il source file has either trailing white space or an end-of-line character.

Option	Description
<code>/32bitpreferred</code>	Creates a 32-bit-preferred image (PE32).
<code>/alignment:</code> <code>integer</code>	Sets FileAlignment to the value specified by <code>integer</code> in the NT Optional header. If the .alignment IL directive is specified in the file, this option overrides it.
<code>/appcontainer</code>	Produces a <code>.dll</code> or <code>.exe</code> file that runs in the Windows app container, as output.
<code>/arm</code>	Specifies the Advanced RISC Machine (ARM) as the target processor. If no image bitness is specified, the default is <code>/32bitpreferred</code> .

Option	Description
<code>/base: integer</code>	Sets ImageBase to the value specified by <code>integer</code> in the NT Optional header. If the .imagebase IL directive is specified in the file, this option overrides it.
<code>/clock</code>	Measures and reports the following compilation times in milliseconds for the specified .il source file:
	<p>Total Run: The total time spent performing all the specific operations that follow.</p>
	<p>Startup: Loading and opening the file.</p>
	<p>Emitting MD: Emitting metadata.</p>
	<p>Ref to Def Resolution: Resolving references to definitions in the file.</p>
	<p>CEE File Generation: Generating the file image in memory.</p>
	<p>PE File Writing: Writing the image to a PE file.</p>
<code>/debug[:IMPL OPT]</code>	Includes debug information (local variable and argument names, and line numbers). Creates a PDB file.
	<p><code>/debug</code> with no additional value disables JIT optimization and uses sequence points from the PDB file.</p>
	<p><code>IMPL</code> disables JIT optimization and uses implicit sequence points.</p>
	<p><code>OPT</code> enables JIT optimization and uses implicit sequence points.</p>
<code>/dll</code>	Produces a <code>.dll</code> file as output.
<code>/enc: file</code>	<p>Creates Edit-and-Continue deltas from the specified source file.</p> <p>This argument is for academic use only and is not supported for commercial use.</p>
<code>/exe</code>	Produces an executable file as output. This is the default.
<code>/flags: integer</code>	Sets ImageFlags to the value specified by <code>integer</code> in the common language runtime header. If the .corflags IL directive is specified in the file, this option overrides it. See CorHdr.h, COMIMAGE_FLAGS for a list of valid values for <code>integer</code> .
<code>/fold</code>	Folds identical method bodies into one.
<code>/highentropyva</code>	Produces an output executable that supports high-entropy address space layout randomization (ASLR). (Default for <code>/appcontainer</code> .)

Option	Description
/include: includePath	Sets a path to search for files included with <code>#include</code> .
/itanium	Specifies Intel Itanium as the target processor. If no image bitness is specified, the default is <code>/pe64</code> .
/key: keyFile	Compiles <code>filename</code> with a strong signature using the private key contained in <code>keyFile</code> .
/key: @keySource	Compiles <code>filename</code> with a strong signature using the private key produced at <code>keySource</code> .
/listing	Produces a listing file on the standard output. If you omit this option, no listing file is produced. This parameter is not supported in the .NET Framework 2.0 or later.
/mdv: versionString	Sets the metadata version string.
/msv: major.minor	Sets the metadata stream version, where <code>major</code> and <code>minor</code> are integers.
/noautoinherit	Disables default inheritance from Object when no base class is specified.
/nocorstub	Suppresses generation of the CORExeMain stub.
/nologo	Suppresses the Microsoft startup banner display.
/output: file.ext	Specifies the output file name and extension. By default, the output file name is the same as the name of the first source file. The default extension is <code>.exe</code> . If you specify the <code>/dll</code> option, the default extension is <code>.dll</code> . Note: Specifying <code>/output:myfile.dll</code> does not set the <code>/dll</code> option. If you do not specify <code>/dll</code> , the result will be an executable file named <code>myfile.dll</code> .
/optimize	Optimizes long instructions to short. For example, <code>br</code> to <code>br.s</code> .
/pe64	Creates a 64-bit image (PE32+). If no target processor is specified, the default is <code>/itanium</code> .
/pdb	Creates a PDB file without enabling debug information tracking.
/quiet	Specifies quiet mode; does not report assembly progress.
/resource: file.res	Includes the specified resource file in <code>*.res</code> format in the resulting <code>.exe</code> or <code>.dll</code> file. Only one <code>.res</code> file can be specified with the <code>/resource</code> option.

Option	Description
/ssver: <code>int.int</code>	Sets the subsystem version number in the NT optional header. For <code>/appcontainer</code> and <code>/arm</code> the minimum version number is 6.02.
/stack: <code>stackSize</code>	Sets the <code>SizeOfStackReserve</code> value in the NT Optional header to <code>stackSize</code> .
/strip reloc	Specifies that no base relocations are needed.
/subsystem: <code>integer</code>	Sets subsystem to the value specified by <code>integer</code> in the NT Optional header. If the <code>.subsystem</code> IL directive is specified in the file, this command overrides it. See <code>winnt.h</code> , <code>IMAGE_SUBSYSTEM</code> for a list of valid values for <code>integer</code> .
/x64	Specifies a 64-bit AMD processor as the target processor. If no image bitness is specified, the default is <code>/pe64</code> .
/?	Displays command syntax and options for the tool.

ⓘ Note

All options for `Ilasm.exe` are case-insensitive and recognized by the first three letters. For example, `/lis` is equivalent to `/listing` and `/res:myresfile.res` is equivalent to `/resource:myresfile.res`. Options that specify arguments accept either a colon (:) or an equal sign (=) as the separator between the option and the argument. For example, `/output:file.ext` is equivalent to `/output=file.ext`.

Remarks

The IL Assembler helps tool vendors design and implement IL generators. Using `Ilasm.exe`, tool and compiler developers can concentrate on IL and metadata generation without being concerned with emitting IL in the PE file format.

Similar to other compilers that target the runtime, such as C# and Visual Basic, `Ilasm.exe` does not produce intermediate object files and does not require a linking stage to form a PE file.

The IL Assembler can express all the existing metadata and IL features of the programming languages that target the runtime. This allows managed code written in any of these programming languages to be adequately expressed in IL Assembler and compiled with `Ilasm.exe`.

ⓘ Note

Compilation might fail if the last line of code in the .il source file does not have either trailing white space or an end-of-line character.

You can use *Ilasm.exe* in conjunction with its companion tool, *Ildasm.exe*. *Ildasm.exe* takes a PE file that contains IL code and creates a text file suitable as input to *Ilasm.exe*. This is useful, for example, when compiling code in a programming language that does not support all the runtime metadata attributes. After compiling the code and running the output through *Ildasm.exe*, the resulting IL text file can be hand-edited to add the missing attributes. You can then run this text file through the *Ilasm.exe* to produce a final executable file.

You can also use this technique to produce a single PE file from several PE files originally generated by different compilers.

 **Note**

Currently, you cannot use this technique with PE files that contain embedded native code (for example, PE files produced by Visual C++).

To make this combined use of *Ildasm.exe* and *Ilasm.exe* as accurate as possible, by default the assembler does not substitute short encodings for long ones you might have written in your IL sources (or that might be emitted by another compiler). Use the **/optimize** option to substitute short encodings wherever possible.

 **Note**

Ildasm.exe only operates on files on disk. It does not operate on files installed in the global assembly cache.

For more information about the grammar of IL, see the `asmparse.grammar` file in the Windows SDK.

Version Information

Starting with .NET Framework 4.5, you can attach a custom attribute to an interface implementation by using code similar to the following:

```
il
```

```
.class interface public abstract auto ansi IMyInterface
{
```

```
.method public hidebysig newslot abstract virtual
    instance int32 method1() cil managed
{
} // end of method IMyInterface::method1
} // end of class IMyInterface
.class public auto ansi beforefieldinit MyClass
    extends [mscorlib]System.Object
    implements IMyInterface
{
    .interfaceimpl type IMyInterface
    .custom instance void
        [mscorlib]System.Diagnostics.DebuggerNonUserCodeAttribute::.ctor() = (
01 00 00 00 )
    ...
}
```

Starting with .NET Framework 4.5, you can specify an arbitrary marshal BLOB (binary large object) by using its raw binary representation, as shown in the following code:

```
il

.method public hidebysig abstract virtual
    instance void
    marshal({ 38 01 02 FF })
    Test(object A_1) cil managed
```

For more information about the grammar of IL, see the `asmparse.grammar` file in the Windows SDK.

Examples

The following command assembles the IL file `myTestFile.il` and produces the executable `myTestFile.exe`.

```
Console

ilasm myTestFile
```

The following command assembles the IL file `myTestFile.il` and produces the `.dll` file `myTestFile.dll`.

```
Console

ilasm myTestFile /dll
```

The following command assembles the IL file *myTestFile.il* and produces the *.dll* file *myNewTestFile.dll*.

```
Console
```

```
ilasm myTestFile /dll /output:myNewTestFile.dll
```

The following code example shows an extremely simple application that displays "Hello World!" to the console. You can compile this code and then use the *Ildasm.exe* tool to generate an IL file.

```
C#
```

```
using System;

public class Hello
{
    public static void Main(String[] args)
    {
        Console.WriteLine("Hello World!");
    }
}
```

The following IL code example corresponds to the previous C# code example. You can compile this code into an assembly using the IL Assembler tool. Both IL and C# code examples display "Hello World!" to the console.

```
il
```

```
// Metadata version: v2.0.50215
.assembly extern mscorel
{
    .publickeytoken = (B7 7A 5C 56 19 34 E0 89 ) //z\V.4..
    .ver 2:0:0:0
}
.assembly sample
{
    .custom instance void
[mscorel]System.Runtime.CompilerServices.CompilationRelaxationsAttribute::ctor(int32) = ( 01 00 08 00 00 00 00 00 )
    .hash algorithm 0x00008004
    .ver 0:0:0:0
}
.module sample.exe
// MVID: {A224F460-A049-4A03-9E71-80A36DBBCD3}
.imagebase 0x00400000
.file alignment 0x00000200
.stackreserve 0x00100000
```

```

.subsystem 0x0003      // WINDOWS_CUI
.corflags 0x00000001    // ILOONLY
// Image base: 0x02F20000

// ===== CLASS MEMBERS DECLARATION =====

.class public auto ansi beforefieldinit Hello
    extends [mscorlib]System.Object
{
    .method public hidebysig static void Main(string[] args) cil managed
    {
        .entrypoint
        // Code size      13 (0xd)
        .maxstack 8
        IL_0000:  nop
        IL_0001:  ldstr     "Hello World!"
        IL_0006:  call       void [mscorlib]System.Console::WriteLine(string)
        IL_000b:  nop
        IL_000c:  ret
    } // end of method Hello::Main

    .method public hidebysig specialname rtspecialname
        instance void .ctor() cil managed
    {
        // Code size      7 (0x7)
        .maxstack 8
        IL_0000:  ldarg.0
        IL_0001:  call       instance void [mscorlib]System.Object::ctor()
        IL_0006:  ret
    } // end of method Hello::ctor
}

} // end of class Hello

```

See also

- [Tools](#)
- [*Ildasm.exe* \(IL Disassembler\)](#)
- [Managed Execution Process](#)
- [Developer command-line shells](#)

Ildasm.exe (IL Disassembler)

Article • 07/23/2022

The IL Disassembler is a companion tool to the IL Assembler (*Ilasm.exe*). *Ildasm.exe* takes a portable executable (PE) file that contains intermediate language (IL) code and creates a text file suitable as input to *Ilasm.exe*.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
ildasm [options] [PEfilename] [options]
```

Parameters

The following options are available for *.exe*, *.dll*, *.obj*, *.lib*, and *.winmd* files.

Option	Description
/out= <i>filename</i>	Creates an output file with the specified <i>filename</i> , rather than displaying the results in a graphical user interface.
/rtf	Produces output in rich text format. Invalid with the <i>/text</i> option.
/text	Displays the results to the console window, rather than in a graphical user interface or as an output file.
/html	Produces output in HTML format. Valid with the <i>/output</i> option only.
/?	Displays the command syntax and options for the tool.

The following additional options are available for *.exe*, *.dll*, and *.winmd* files.

Option	Description
/bytes	Shows actual bytes, in hexadecimal format, as instruction comments.
/caverbal	Produces custom attribute blobs in verbal form. The default is binary form.

Option	Description
/linenum	Includes references to original source lines.
/nobar	Suppresses the disassembly progress indicator pop-up window.
/noca	Suppresses the output of custom attributes.
/project	Displays metadata the way it appears to managed code, instead of the way it appears in the native Windows Runtime. If <code>PEfilename</code> is not a Windows metadata (.winmd) file, this option has no effect. See .NET Framework Support for Windows Store Apps and Windows Runtime .
/pubonly	Disassembles only public types and members. Equivalent to <code>/visibility:PUB</code> .
/quoteallnames	Includes all names in single quotes.
/raweh	Shows exception handling clauses in raw form.
/source	Shows original source lines as comments.
/tokens	Shows metadata tokens of classes and members.
/visibility: vis [+ vis ...]	Disassembles only types or members with the specified visibility. The following are valid values for <code>vis</code> :
	PUB — Public
	PRI — Private
	FAM — Family
	ASM — Assembly
	FAA — Family and Assembly
	FOA — Family or Assembly
	PSC — Private Scope
	For definitions of these visibility modifiers, see MethodAttributes and TypeAttributes .

The following options are valid for `.exe`, `.dll`, and `.winmd` files for file or console output only.

Option	Description
/all	Specifies a combination of the <code>/header</code> , <code>/bytes</code> , <code>/stats</code> , <code>/classlist</code> , and <code>/tokens</code> options.

Option	Description
/classlist	Includes a list of classes defined in the module.
/forward	Uses forward class declaration.
/headers	Includes file header information in the output.
/item: class [:: member [(sig)]]	<p>Disassembles the following depending upon the argument supplied:</p> <ul style="list-style-type: none"> - Disassembles the specified <code>class</code>. - Disassembles the specified <code>member</code> of the <code>class</code>. - Disassembles the <code>member</code> of the <code>class</code> with the specified signature <code>sig</code>. The format of <code>sig</code> is: <p style="padding-left: 20px;">[instance] returnType(parameterType1, parameterType2, ..., parameterTypeN)</p> <p>Note In the .NET Framework versions 1.0 and 1.1, <code>sig</code> must be followed by a closing parenthesis: <code>(sig)</code>. Starting with the Net Framework 2.0 the closing parenthesis must be omitted: <code>(sig)</code>.</p>
/noil	Suppresses IL assembly code output.
/stats	Includes statistics on the image.
/typelist	Produces the full list of types, to preserve type ordering in a round trip.
/unicode	Uses Unicode encoding for the output.
/utf8	Uses UTF-8 encoding for the output. ANSI is the default.

The following options are valid for `.exe`, `.dll`, `.obj`, `.lib`, and `.winmd` files for file or console output only.

Option	Description

Option	Description
/metadata[=specifier]	<p>Shows metadata, where <code>specifier</code> is:</p> <ul style="list-style-type: none"> MDHEADER — Show the metadata header information and sizes. HEX — Show information in hex as well as in words. CSV — Show the record counts and heap sizes. UNREX — Show unresolved externals. SCHEMA — Show the metadata header and schema information. RAW — Show the raw metadata tables. HEAPS — Show the raw heaps. VALIDATE — Validate the consistency of the metadata. <p>You can specify <code>/metadata</code> multiple times, with different values for <code>specifier</code>.</p>

The following options are valid for `.lib` files for file or console output only.

Option	Description
/objectfile=filename	Shows the metadata of a single object file in the specified library.

① Note

All options for `Ildasm.exe` are case-insensitive and recognized by the first three letters. For example, `/quo` is equivalent to `/quoteallnames`. Options that specify arguments accept either a colon (`:`) or an equal sign (`=`) as the separator between the option and the argument. For example, `/output:filename` is equivalent to `/output=filename`.

Remarks

`Ildasm.exe` only operates on PE files on disk. It does not operate on files installed in the global assembly cache.

The text file produced by `Ildasm.exe` can be used as input to the IL Assembler (`Ilasm.exe`). This is useful, for example, when compiling code in a programming language that does

not support all the runtime metadata attributes. After compiling the code and running its output through *Ildasm.exe*, the resulting IL text file can be hand-edited to add the missing attributes. You can then run this text file through the IL Assembler to produce a final executable file.

 **Note**

Currently, you cannot use this technique with PE files that contain embedded native code (for example, PE files produced by Visual C++).

You can use the default GUI in the IL Disassembler to view the metadata and disassembled code of any existing PE file in a hierarchical tree view. To use the GUI, type **ildasm** at the command line without supplying the *PEfilename* argument or any options. From the **File** menu, you can navigate to the PE file that you want to load into *Ildasm.exe*. To save the metadata and disassembled code displayed for the selected PE, select the **Dump** command from the **File** menu. To save the hierarchical tree view only, select the **Dump Treeview** command from the **File** menu. For a detailed guide to loading a file into *Ildasm.exe* and interpreting the output, see the *Ildasm.exe* Tutorial, located in the Samples folder that ships with the Windows SDK.

If you provide *Ildasm.exe* with a *PEfilename* argument that contains embedded resources, the tool produces multiple output files: a text file that contains IL code and, for each embedded managed resource, a .resources file produced using the resource's name from metadata. If an unmanaged resource is embedded in *PEfilename*, a .res file is produced using the filename specified for IL output by the **/output** option.

 **Note**

Ildasm.exe shows only metadata descriptions for *.obj* and *.lib* input files. IL code for these file types is not disassembled.

You can run *Ildasm.exe* over an.exe or .dll file to determine whether the file is managed. If the file is not managed, the tool displays a message stating that the file has no valid common language runtime header and cannot be disassembled. If the file is managed, the tool runs successfully.

Version Information

Starting with .NET Framework 4.5, *Ildasm.exe* handles an unrecognized marshal BLOB (binary large object) by displaying the raw binary content. For example, the following

code shows how a marshal BLOB generated by a C# program is displayed:

C#

```
public void Test([MarshalAs((short)70)] int test) { }
```

il

```
// IL from Ildasm.exe output
.method public hidebysig instance void Test(int32 marshal({ 46 }) test) cil
managed
```

Starting with .NET Framework 4.5, *Ildasm.exe* displays attributes that are applied to interface implementations, as shown in the following excerpt from *Ildasm.exe* output:

il

```
.class public auto ansi beforefieldinit MyClass
  extends [mscorlib]System.Object
  implements IMyInterface
{
  .interfaceimpl type IMyInterface
  .custom instance void
    [mscorlib]System.Diagnostics.DebuggerNonUserCodeAttribute::ctor() = (
  01 00 00 00 )
  ...
}
```

Examples

The following command causes the metadata and disassembled code for the PE file `MyHello.exe` to display in the *Ildasm.exe* default GUI.

Console

```
ildasm myHello.exe
```

The following command disassembles the file `MyFile.exe` and stores the resulting IL Assembler text in the file `MyFile.il`.

Console

```
ildasm MyFile.exe /output:MyFile.il
```

The following command disassembles the file `MyFile.exe` and displays the resulting IL Assembler text to the console window.

```
Console
```

```
ildasm MyFile.exe /text
```

If the file `MyApp.exe` contains embedded managed and unmanaged resources, the following command produces four files: `MyApp.il`, `MyApp.res`, `Icons.resources`, and `Message.resources`:

```
Console
```

```
ildasm MyApp.exe /output:MyApp.il
```

The following command disassembles the method `MyMethod` within the class `MyClass` in `MyFile.exe` and displays the output to the console window.

```
Console
```

```
ildasm /item:MyClass::MyMethod MyFile.exe /text
```

In the previous example, there could be several methods named `MyMethod` with different signatures. The following command disassembles the instance method `MyMethod` with the return type of `void` and the parameter types `int32` and `string`.

```
Console
```

```
ildasm /item:"MyClass::MyMethod(instance void(int32,string))" MyFile.exe  
/text
```

ⓘ Note

In the .NET Framework versions 1.0 and 1.1, the left parenthesis that follows the method name must be balanced by a right parenthesis after the signature:
`MyMethod(instance void(int32))`. Starting with the .NET Framework 2.0 the closing parenthesis must be omitted: `MyMethod(instance void(int32))`.

To retrieve a `static` method (`Shared` method in Visual Basic), omit the keyword `instance`. Class types that are not primitive types like `int32` and `string` must include the namespace and must be preceded by the keyword `class`. External types must be

preceded by the library name in square brackets. The following command disassembles a static method named `MyMethod` that has one parameter of type `AppDomain` and has a return type of `AppDomain`.

```
Console  
  
ildasm /item:"MyClass::MyMethod(class [mscorlib]System.AppDomain(class  
[mscorlib]System.AppDomain)" MyFile.exe /text
```

A nested type must be preceded by its containing class, delimited by a forward slash. For example, if the `MyNamespace.MyClass` class contains a nested class named `NestedClass`, the nested class is identified as follows: `class MyNamespace.MyClass/NestedClass`.

See also

- [Tools](#)
- [Ilasm.exe \(IL Assembler\)](#)
- [Managed Execution Process](#)
- [Developer command-line shells](#)

Installutil.exe (Installer tool)

Article • 07/23/2022

The Installer tool is a command-line utility that allows you to install and uninstall server resources by executing the installer components in specified assemblies. This tool works in conjunction with classes in the [System.Configuration.Install](#) namespace.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
installutil [/u[ninstall]] [options] assembly [[options] assembly] ...
```

Parameters

[+] Expand table

Argument	Description
<code>assembly</code>	The file name of the assembly in which to execute the installer components. Omit this parameter if you want to specify the assembly's strong name by using the <code>/AssemblyName</code> option.

Options

[+] Expand table

Option	Description
<code>/h[elp]</code>	Displays command syntax and options for the tool.
-or-	
<code>/?</code>	

Option	Description
<pre>/help assembly -or- /? assembly</pre>	<p>Displays additional options recognized by individual installers within the specified assembly, along with command syntax and options for InstallUtil.exe. This option adds the text returned by each installer component's Installer.HelpText property to the help text of InstallUtil.exe. For example, if ServiceProcessInstaller.Account is <code>User</code>, the <code>/username</code> and <code>/password</code> options are available.</p>
<pre>/AssemblyName "assemblyName" ,Version=major.minor.build.revision ,Culture=locale ,PublicKeyToken=publicKeyToken"</pre>	<p>Specifies the strong name of an assembly, which must be registered in the global assembly cache. The assembly name must be fully qualified with the version, culture, and public key token of the assembly. The fully qualified name must be surrounded by quotes.</p> <p>For example, "myAssembly, Culture=neutral, PublicKeyToken=0038abc9deabfle5, Version=4.0.0.0" is a fully qualified assembly name.</p>
<pre>/InstallStateDir=[directoryName]</pre>	<p>Specifies the directory of the .InstallState file that contains the data used to uninstall the assembly. The default is the directory that contains the assembly.</p>
<pre>/LogFile=[filename]</pre>	<p>Specifies the name of the log file where installation progress is recorded. By default, if the <code>/LogFile</code> option is omitted, a log file named <i>assemblyname</i>.InstallLog is created. If <i>filename</i> is omitted, no log file is generated.</p>
<pre>/LogToConsole={true false}</pre>	<p>If <code>true</code>, displays output to the console. If <code>false</code> (the default), suppresses output to the console.</p>
<pre>/ShowCallStack</pre>	<p>Outputs the call stack to the log file if an exception occurs at any point during installation.</p>
<pre>/u[ninstall]</pre>	<p>Uninstalls the specified assemblies. Unlike the other options, <code>/u</code> applies to all assemblies regardless of where the option appears on the command line.</p>

Additional installer options

Individual installers used within an assembly may recognize options in addition to those listed in the [Options](#) section. To learn about these options, run InstallUtil.exe with the paths of the assemblies on the command line along with the `/?` or `/help` option. To specify these options, you include them on the command line along with the options recognized by InstallUtil.exe.

Note

Help text on the options supported by individual installer components is returned by the [Installer.HelpText](#) property. The individual options that have been entered on the command line are accessible programmatically from the [Installer.Context](#) property.

All options and command-line parameters are written to the installation log file. However, if you use the `/Password` parameter, which is recognized by some installer components, the password information is replaced by eight asterisks (*) and won't appear in the log file.

Important

In some cases, parameters passed to the installer may include sensitive or personally identifiable information, which, by default, is written to a plain text log file. To prevent this behavior, you can suppress the log file by specifying `/LogFile=` (with no *filename* argument) on the command line.

Remarks

.NET Framework applications consist of traditional program files and associated resources, such as message queues, event logs, and performance counters, that must be created when the application is deployed. You can use an assembly's installer components to create these resources when your application is installed and to remove them when your application is uninstalled. Installutil.exe detects and executes these installer components.

You can specify multiple assemblies on the same command line. Any option that occurs before an assembly name applies to that assembly's installation. Except for `/u` and `/AssemblyName`, options are cumulative but overridable. That is, options specified for one assembly apply to all subsequent assemblies unless the option is specified with a new value.

If you run Installutil.exe against an assembly without specifying any options, it places the following three files into the assembly's directory:

- `InstallUtil.InstallLog` - Contains a general description of the installation progress.
- `assemblyname.InstallLog` - Contains information specific to the commit phase of the installation process. For more information about the commit phase, see the

[Commit](#) method.

- `assemblyname.InstallState` - Contains data used to uninstall the assembly.

Installutil.exe uses reflection to inspect the specified assemblies and to find all [Installer](#) types that have the [System.ComponentModel.RunInstallerAttribute](#) attribute set to `true`. The tool then executes either the [Installer.Install](#) or the [Installer.Uninstall](#) method on each instance of the [Installer](#) type. Installutil.exe performs installation in a transactional manner; that is, if one of the assemblies fails to install, it rolls back the installations of all other assemblies. Uninstall is not transactional.

Installutil.exe cannot install or uninstall delay-signed assemblies, but it can install or uninstall strong-named assemblies.

The 32-bit version of the common language runtime (CLR) ships with only the 32-bit version of the Installer tool, but the 64-bit version of the CLR ships with both 32-bit and 64-bit versions of the Installer tool. When using the 64-bit CLR, use the 32-bit Installer tool to install 32-bit assemblies, and the 64-bit Installer tool to install 64-bit and common intermediate language (CIL) assemblies. Both versions of the Installer tool behave the same.

You can't use Installutil.exe to deploy a Windows service that was created by using C++, because Installutil.exe doesn't recognize the embedded native code that's produced by the C++ compiler. If you try to deploy a C++ Windows service with Installutil.exe, an exception such as [BadImageFormatException](#) will be thrown. To work with this scenario, move the service code to a C++ module, and then write the installer object in C# or Visual Basic.

Examples

The following command displays a description of the command syntax and options for InstallUtil.exe.

```
Console
```

```
installutil /?
```

The following command displays a description of the command syntax and options for InstallUtil.exe. It also displays a description and list of options supported by the installer components in `myAssembly.exe` if help text has been assigned to the installer's [Installer.HelpText](#) property.

```
Console
```

```
installutil /? myAssembly.exe
```

The following command executes the installer components in the assembly `myAssembly.exe`.

Console

```
installutil myAssembly.exe
```

The following command executes the installer components in an assembly by using the `/AssemblyName` switch and a fully qualified name.

Console

```
installutil /AssemblyName "myAssembly, Culture=neutral,  
PublicKeyToken=0038abc9deabfle5, Version=4.0.0.0"
```

The following command executes the installer components in an assembly specified by file name and in an assembly specified by strong name. Note that all assemblies specified by file name must precede assemblies specified by strong name on the command line, because the `/AssemblyName` option cannot be overridden.

Console

```
installutil myAssembly.exe /AssemblyName "myAssembly, Culture=neutral,  
PublicKeyToken=0038abc9deabfle5, Version=4.0.0.0"
```

The following command executes the uninstaller components in the assembly `myAssembly.exe`.

Console

```
installutil /u myAssembly.exe
```

The following command executes the uninstaller components in the assemblies `myAssembly1.exe` and `myAssembly2.exe`.

Console

```
installutil myAssembly1.exe /u myAssembly2.exe
```

Because the position of the `/u` option on the command line is not important, this is equivalent to the following command.

Console

```
installutil /u myAssembly1.exe myAssembly2.exe
```

The following command executes the installers in the assembly `myAssembly.exe` and specifies that progress information will be written to `myLog.InstallLog`.

Console

```
installutil /LogFile=myLog.InstallLog myAssembly.exe
```

The following command executes the installers in the assembly `myAssembly.exe`, specifies that progress information should be written to `myLog.InstallLog`, and uses the installers' custom `/reg` option to specify that updates should be made to the system registry.

Console

```
installutil /LogFile=myLog.InstallLog /reg=true myAssembly.exe
```

The following command executes the installers in the assembly `myAssembly.exe`, uses the installer's custom `/email` option to specify the user's email address, and suppresses output to the log file.

Console

```
installutil /LogFile= /email=admin@mycompany.com myAssembly.exe
```

The following command writes the installation progress for `myAssembly.exe` to `myLog.InstallLog` and writes the progress for `myTestAssembly.exe` to `myTestLog.InstallLog`.

Console

```
installutil /LogFile=myLog.InstallLog myAssembly.exe  
/LogFile=myTestLog.InstallLog myTestAssembly.exe
```

See also

- [System.Configuration.Install](#)
- [Tools](#)
- [Developer command-line shells](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Lc.exe (License Compiler)

Article • 09/15/2021

The License Compiler reads text files that contain licensing information and produces a binary file that can be embedded in a common language runtime executable as a resource.

A .licx text file is automatically generated or updated by the Windows Forms Designer whenever a licensed control is added to the form. As part of compilation, the project system will transform the .licx text file into a .licenses binary resource that provides support for .NET control licensing. The binary resource will then be embedded in the project output.

Cross compilation between 32-bit and 64-bit is not supported when you use the License Compiler when building your project. This is because the License Compiler has to load assemblies, and loading 64-bit assemblies from a 32-bit application is not allowed, and vice versa. In this case, use the License Compiler from the command line to compile the license manually, and specify the corresponding architecture.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
lc /target:  
targetPE /complist:filename [-outdir:path]  
/i:modules [/nologo] [/v]
```

Option	Description
/complist: <i>filename</i>	Specifies the name of a file that contains the list of licensed components to include in the .licenses file. Each component is referenced using its full name with only one component per line. Command-line users can specify a separate file for each form in the project. Lc.exe accepts multiple input files and produces a single .licenses file.
/h[elp]	Displays command syntax and options for the tool.

Option	Description
/i: <i>module</i>	Specifies the modules that contain the components listed in the /complist file. To specify more than one module, use multiple /i flags.
/nologo	Suppresses the Microsoft startup banner display.
/outdir: <i>path</i>	Specifies the directory in which to place the output .licenses file.
/target: <i>targetPE</i>	Specifies the executable for which the .licenses file is being generated.
/v	Specifies verbose mode; displays compilation progress information.
@ <i>file</i>	Specifies the response (.rsp) file.
/?	Displays command syntax and options for the tool.

Example

- If you are using a licensed control `MyCompany.Samples.LicControl1` contained in `Samples.DLL` in an application called `HostApp.exe`, you can create `HostAppLic.txt` that contains the following.

```
text
MyCompany.Samples.LicControl1, Samples.DLL
```

- Create the .licenses file called `HostApp.exe.licenses` using the following command.

```
Console
lc /target:HostApp.exe /complist:hostapplic.txt /i:Samples.DLL
/outdir:c:\bindir
```

- Build `HostApp.exe` including the .licenses file as a resource. If you were building a C# application you would use the following command to build your application.

```
Console
csc /res:HostApp.exe.licenses /out:HostApp.exe *.cs
```

The following command compiles `myApp.licenses` from the lists of licensed components specified by `hostapplic.txt`, `hostapplic2.txt` and `hostapplic3.txt`. The `modulesList`

argument specifies the modules that contain the licensed components.

Console

```
lc /target:myApp /complist:hostapplic.txt /complist:hostapplic2.txt  
/complist: hostapplic3.txt /i:modulesList
```

Response File Example

The following listing shows an example of a response file, `response.rsp`. For more information on response files, see [Response Files](#).

text

```
/target:hostapp.exe  
/complist:hostapplic.txt  
/i:WFCPrj.dll  
/outdir:"C:\My Folder"
```

The following command line uses the `response.rsp` file.

Console

```
lc @response.rsp
```

See also

- [Tools](#)
- [Al.exe \(Assembly Linker\)](#)
- [Developer command-line shells](#)

Mage.exe (Manifest Generation and Editing Tool)

Article • 03/30/2023

The Manifest Generation and Editing Tool (*Mage.exe*) is a command-line tool that supports the creation and editing of application and deployment manifests. As a command-line tool, *Mage.exe* can be run from both batch scripts and other Windows-based applications, including ASP.NET applications.

You can also use *MageUI.exe*, a graphical application, instead of *Mage.exe*. For more information, see [MageUI.exe \(Manifest Generation and Editing Tool, Graphical Client\)](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

Two versions of *Mage.exe* and *MageUI.exe* are included with Visual Studio. To see version information, run *MageUI.exe*, select **Help**, and select **About**. This documentation describes version 4.0.x.x of *Mage.exe* and *MageUI.exe*.

Syntax

Console

```
Mage [commands] [commandOptions]
```

Parameters

The following table shows the commands supported by *Mage.exe*. For more information about the options supported by these commands, see [New and Update command options](#) and [Sign command options](#).

[+] Expand table

Command	Description
<code>-cc,</code> <code>ClearApplicationCache</code>	Clears the downloaded application cache of all online-only applications.
<code>-n, -New fileType</code> <code>[newOptions]</code>	Creates a new file of the given type. Valid types are: - Deployment : Creates a new deployment manifest.

Command	Description
	<p>- Application: Creates a new application manifest.</p> <p>If you do not specify any additional parameters with this command, it will create a file of the appropriate type, with appropriate default tags and attribute values.</p> <p>Use the -ToFile option (see in the following table) to specify the file name and path of the new file.</p> <p>Use the -FromDirectory option (see in the following table) to create an application manifest with all of the assemblies for an application added to the <dependency> section of the manifest.</p>
-u, -Update [<i>filePath</i>] [<i>updateOptions</i>]	<p>Makes one or more changes to a manifest file. You do not have to specify the type of file that you are editing. Mage.exe will examine the file by using a set of heuristics and determine whether it is a deployment manifest or an application manifest.</p> <p>If you have already signed a file with a certificate, -Update will remove the key signature block. This is because the key signature contains a hash of the file, and modifying the file renders the hash invalid.</p> <p>Use the -ToFile option (see in the following table) to specify a new file name and path instead of overwriting the existing file.</p>
-s, -Sign [<i>signOptions</i>]	<p>Uses a key pair or X509 certificate to sign a file. Signatures are inserted as XML elements inside of the files.</p> <p>You must be connected to the Internet when signing a manifest that specifies a -TimestampUri value.</p>
-ver, -Verify [<i>manifest-filename</i>]	<p>Verifies that the manifest is signed correctly. Cannot be combined with other commands.</p> <p>Available in .NET Framework 4.7 and later versions.</p>
-h, -?, -Help [<i>verbose</i>]	<p>Describes all of the available commands and their options. Specify verbose to get detailed help.</p>

New and Update command options

The following table shows the options supported by the **-New** and **-Update** commands:

[] Expand table

Options	Default Value	Applies To	Description
-a, -Algorithm	sha1RSA	Application manifests. Deployment manifests.	Specifies the algorithm to generate dependency digests with. Value must be "sha256RSA" or "sha1RSA". Use with the "-Update" option. This option is ignored when using the "-Sign" option
-appc, -AppCodeBase <code>manifestReference</code>		Deployment manifests.	Inserts a URL or file path reference to the application manifest file. This value must be the full path to the application manifest.
-appm, -AppManifest <code>manifestPath</code>		Deployment manifests.	Inserts a reference to a deployment's application manifest into its deployment manifest. The file indicated by <code>manifestPath</code> must exist, or <i>Mage.exe</i> will issue an error. If the file referenced by <code>manifestPath</code> is not an application manifest, <i>Mage.exe</i> will issue an error.
-cf, -CertFile <code>filePath</code>	All file types.		Specifies the location of an X509 digital certificate for signing a manifest or license file. This option can be used in conjunction with the -Password option if the certificate requires a password for Personal Information Exchange (PFX) files. Starting with .NET Framework 4.7, if the file does not contain a private key, a combination of the -CryptoProvider and -KeyContainer options is required. Starting with .NET Framework 4.6.2, <i>Mage.exe</i> signs manifests

Options	Default Value	Applies To	Description
			with CNG as well as CAPI certificates.
-ch, -CertHash <code>hashSignature</code>		All file types.	<p>The hash of a digital certificate stored in the personal certificate store of the client computer. This corresponds to the Thumbprint string of a digital certificate viewed in the Windows Certificates Console.</p> <p><code>hashSignature</code> can be either uppercase or lowercase, and can be supplied either as a single string, or with each octet of the Thumbprint separated by spaces and the entire Thumbprint enclosed in quotation marks.</p>
-csp, -CryptoProvider <code>provider-name</code>		All file types.	<p>Specifies the name of a cryptographic service provider (CSP) that contains the private key container. This option requires the -KeyContainer option.</p> <p>This option is available starting with .NET Framework 4.7.</p>
-fd, -FromDirectory <code>directoryPath</code>		Application manifests.	<p>Populates the application manifest with descriptions of all assemblies and files found in <code>directoryPath</code>, including all subdirectories, where <code>directoryPath</code> is the directory that contains the application that you want to deploy. For each file in the directory, <i>Mage.exe</i> decides whether the file is an assembly or a static file. If it is an assembly, it adds a <code><dependency></code> tag and <code>installFrom</code> attribute to the application with the assembly's name, code base, and version. If it is a static file, it adds a <code><file></code> tag. <i>Mage.exe</i> will also</p>

Options	Default Value	Applies To	Description
			<p>use a simple set of heuristics to detect the main executable for the application, and will mark it as the ClickOnce application's entry point in the manifest.</p> <p><i>Mage.exe</i> will never automatically mark a file as a "data" file. You must do this manually. For more information, see How to: Include a Data File in a ClickOnce Application.</p> <p><i>Mage.exe</i> also generates a hash for each file based on its size. ClickOnce uses these hashes to ensure that no one has tampered with the deployment's files since the manifest was created. If any of the files in your deployment change, you can run <i>Mage.exe</i> with the -Update command and the -FromDirectory option, and it will update the hashes and assembly versions of all referenced files.</p> <p>-FromDirectory will include all files in all subdirectories found within <code>directoryPath</code>.</p> <p>If you use -FromDirectory with the -Update command, <i>Mage.exe</i> will remove any files in the application manifest that no longer exist in the directory.</p>
-if, -IconFile <code>filePath</code>		Application manifests.	Specifies the full path to an .ICO icon file. This icon appears beside your application name in the start menu, and in its Add-or-Remove Programs entry. If no icon is provided, a default icon is used.

Options	Default Value	Applies To	Description
-ip, -IncludeProviderURL [url]	true	Deployment manifests.	Indicates whether the deployment manifest includes the update location value set by -ProviderURL .
-i, -Install [willInstall]	true	Deployment manifests.	Indicates whether or not the ClickOnce application should install onto the local computer, or whether it should run from the Web. Installing an application gives that application a presence in the Windows Start menu. Valid values are "true" or "t", and "false" or "f".
			If you specify the -MinVersion option, and a user has a version less than -MinVersion installed, it will force the application to install, regardless of the value that you pass to -Install .
			This option cannot be used with the -BrowserHosted option. Attempting to specify both for the same manifest will result in an error.
-kc, -KeyContainer [name]	All file types.		Specifies the key container that contains the name of the private key. This option requires the CryptoProvider option.
-mv, -MinVersion [version]	The version listed in the ClickOnce deployment manifest as specified by the -Version flag.	Deployment manifests.	The minimum version of this application a user can run. This flag makes the named version of your application a required update. If you release a version of your product with an update to a breaking change or a critical security flaw, you can use this flag to specify that this

Options	Default Value	Applies To	Description
			<p>update must be installed, and that the user cannot continue to run earlier versions.</p> <p><code>version</code> has the same semantics as the argument to the -Version flag.</p>
-n, -Name <code>nameString</code>	Deploy	All file types.	<p>The name that is used to identify the application. ClickOnce will use this name to identify the application in the Start menu (if the application is configured to install itself) and in Permission Elevation dialog boxes. Note: If you are updating an existing manifest and you do not specify a publisher name with this option, <i>Mage.exe</i> updates the manifest with the organization name defined on the computer. To use a different name, make sure to use this option and specify the desired publisher name.</p>
-pwd, -Password <code>passwd</code>		All file types.	<p>The password that is used for signing a manifest with a digital certificate. Must be used in conjunction with the -CertFile option.</p>
-p, Processor <code>processorValue</code>	Msil	<p>Application manifests.</p> <p>Deployment manifests.</p>	<p>The microprocessor architecture on which this distribution will run. This value is required if you are preparing one or more installations whose assemblies have been precompiled for a specific microprocessor. Valid values include <code>msil</code>, <code>x86</code>, <code>ia64</code>, and <code>amd64</code>. <code>msil</code> is Microsoft intermediate language, which means all of your assemblies are platform-independent, and the common language runtime (CLR) will just-in-time compile</p>

Options	Default Value	Applies To	Description
			them when your application is first run.
-pu, -ProviderURL <code>url</code>		Deployment manifests.	Specifies the URL which ClickOnce will examine for application updates.
-pub, -Publisher <code>publisherName</code>		Application manifests.	Adds the publisher name to the description element of either the deployment or
		Deployment manifests.	application manifest. When used on an application manifest, -UseManifestForTrust must also be specified with a value of "true" or "t"; otherwise, this parameter will raise an error.
-s, -SupportURL <code>url</code>		Application manifests. Deployment manifests.	Specifies the link that appears in Add or Remove Programs for the ClickOnce application.
-ti, -TimestampUri <code>uri</code>		Application manifests. Deployment manifests.	The URL of a digital timestamping service. Timestamping the manifests prevents you from having to re-sign the manifests should your digital certificate expire before you deploy the next version of your application. For more information, see Windows root certificate program members .
-t, -ToFile <code>filePath</code>	<ul style="list-style-type: none"> - New: - Deployment: <code>deploy.application</code> - Application: <code>application.exe.manifest</code> - Update: - The input file. 	All file types.	Specifies the output path of the file that has been created or modified.
			If -ToFile is not supplied when you use -New , the output is written to the current working directory. If -ToFile is not supplied when you use -Update , <code>Mage.exe</code> will write the file back to the input file.

Options	Default Value	Applies To	Description
-tr, -TrustLevel <code>level</code>	Based on the zone in which the application URL resides.	Application manifests.	The level of trust to grant the application on client computers. Values include "Internet", "Intranet", and "FullTrust".
-um, -UseManifestForTrust <code>willUseForTrust</code>	False	Application manifests.	Specifies whether the digital signature of the application manifest will be used for making trust decisions when the application runs on the client. Specifying "true" or "t" indicates that the application manifest will be used for trust decisions. Specifying "false" or "f" indicates that the signature of the deployment manifest will be used.
-v, -Version <code>versionNumber</code>	1.0.0.0	Application manifests. Deployment manifests.	The version of the deployment. The argument must be a valid version string of the format "N.N.N.N", where "N" is an unsigned 32-bit integer.
-wpf, -WPFBrowserApp <code>isWPFApp</code>	false	Application manifests. Deployment manifests.	Use this flag only if the application is a Windows Presentation Foundation (WPF) application that will be hosted inside a browser, and is not a stand-alone executable. Valid values are "true" or "t", and "false" or "f".
			For application manifests, inserts the <code>hostInBrowser</code> attribute under the <code>entryPoint</code> element of the application manifest.
			For deployment manifests, sets the <code>install</code> attribute on the <code>deployment</code> element to false, and saves the deployment manifest with a .xbap extension. Specifying this argument along with the -Install argument produces an

Options	Default Value	Applies To	Description
			error, because a browser-hosted application cannot be an installed, offline application.

Sign command options

The following table shows the options supported by the `-sign` command, which apply to all types of files.

[Expand table](#)

Options	Description
<code>-cf, -CertFile filePath</code>	<p>Specifies The location of a digital certificate for signing a manifest. This option can be used in conjunction with the -Password option if the certificate requires a password for Personal Information Exchange (PFX) files. Starting with .NET Framework 4.7, if the file does not contain a private key, a combination of the -CryptoProvider and -KeyContainer options is required.</p> <p>Starting with .NET Framework 4.6.2, <i>Mage.exe</i> signs manifests with CNG as well as CAPI certificates.</p>
<code>-ch, -CertHash hashSignature</code>	<p>The hash of a digital certificate stored in the personal certificate store of the client computer. This corresponds to the Thumbprint property of a digital certificate viewed in the Windows Certificates Console.</p> <p><code>hashSignature</code> can be either uppercase or lowercase, and can be supplied either as a single string or with each octet of the Thumbprint separated by spaces and the entire Thumbprint enclosed in quotation marks.</p>
<code>-csp, - CryptoProvider provider-name</code>	<p>Specifies the name of a cryptographic service provider (CSP) that contains the private key container. This option requires the -KeyContainer option.</p> <p>This option is available starting with .NET Framework 4.7.</p>
<code>-kc, -KeyContainer name</code>	<p>Specifies the key container that contains the name of the private key. This option requires the CryptoProvider option.</p> <p>This option is available starting with .NET Framework 4.7.</p>
<code>-pwd, -Password passwd</code>	<p>The password that is used for signing a manifest with a digital certificate. Must be used in conjunction with the -CertFile option.</p>
<code>-t, -ToFile filePath</code>	<p>Specifies the output path of the file that has been created or modified.</p>

Remarks

All arguments to *Mage.exe* are case-insensitive. Commands and options can be prefixed with a dash (-) or a forward slash (/).

All of the arguments used with the **-Sign** command can be used at any time with the **-New** or **-Update** commands as well. The following commands are equivalent.

Console

```
mage -Sign c:\HelloWorldDeployment\HelloWorld.deploy -CertFile cert.pfx  
mage -Update c:\HelloWorldDeployment\HelloWorld.deploy -CertFile cert.pfx
```

ⓘ Note

Beginning with .NET Framework version 4.6.2, CNG certificates are also supported.

Signing is the last task you should perform, because a signed document uses a hash of the file to verify that the signature is valid for the document. If you make any changes to a signed file, you must sign it again. If you sign a document that was previously signed, *Mage.exe* will replace the old signature with the new.

When you use the **-AppManifest** option to populate a deployment manifest, *Mage.exe* will assume that your application manifest will reside in the same directory as the deployment manifest within a subdirectory named after the current deployment version, and will configure your deployment manifest appropriately. If your application manifest will reside elsewhere, use the **-AppCodeBase** option to set the alternate location.

Your deployment and application manifest must be signed before you deploy your application. For guidance about signing manifests, see [Trusted Application Deployment Overview](#).

The **-TrustLevel** option for application manifests describes the permission set an application requires to run on the client computer. By default, applications are assigned a trust level based on the zone in which their URL resides. Applications deployed over a corporate network are generally placed in the Intranet zone, while those deployed over the Internet are placed in the Internet zone. Both security zones place restrictions on the application's access to local resources, with the Intranet zone slightly more permissive than the Internet zone. The FullTrust zone gives applications complete access to a computer's local resources. If you use the **-TrustLevel** option to place an application in this zone, the Trust Manager component of the CLR will prompt the user to decide whether they want to grant this higher level of trust. If you are deploying your

application over a corporate network, you can use Trusted Application Deployment to raise the trust level of the application without prompting the user.

Application manifests also support custom trust sections. This helps your application obey the security principle of requesting least permission, as you can configure the manifest to demand only those specific permissions that the application requires in order to execute. *Mage.exe* does not directly support adding a custom trust section. You can add one using a text editor, an XML parser, or the graphical tool *MageUI.exe*. For more information about how to use *MageUI.exe* to add custom trust sections, see [MageUI.exe \(Manifest Generation and Editing Tool, Graphical Client\)](#).

Visual Studio 2017 includes version 4.6.1 of *Mage.exe*. Manifests created with this version of *Mage.exe* target .NET Framework 4. To target older versions of the .NET Framework, use an earlier version of *Mage.exe*.

When you add or remove assemblies from an existing manifest, or re-sign an existing manifest, *Mage.exe* does not update the manifest to target .NET Framework 4.

The following tables show these features and restrictions:

[] Expand table

Manifest version	Operation	Mage v2.0	Mage v4.0
Manifest for applications targeting version 2.0 or 3.x of the .NET Framework	Open	OK	OK
	Close	OK	OK
	Save	OK	OK
	Re-sign	OK	OK
	New	OK	Not supported
	Update (see below)	OK	OK
Manifest for applications targeting version 4 of the .NET Framework	Open	OK	OK
	Close	OK	OK
	Save	OK	OK
	Re-sign	OK	OK

Manifest version	Operation	Mage v2.0	Mage v4.0
	New	Not supported	OK
	Update (see below)	Not supported	OK

[\[+\] Expand table](#)

Manifest version	Update Operation Details	Mage v2.0	Mage v4.0
Manifest for applications targeting version 2.0 or 3.x of the .NET Framework	Modify an assembly	OK	OK
	Add an assembly	OK	OK
	Remove an assembly	OK	OK
Manifest for applications targeting version 4 of the .NET Framework	Modify an assembly	Not supported	OK
	Add an assembly	Not supported	OK
	Remove an assembly	Not supported	OK

Mage.exe creates new manifests that target the .NET Framework 4 Client Profile. ClickOnce applications that target the .NET Framework 4 Client Profile can run on both the .NET Framework 4 Client Profile and the full version of the .NET Framework 4. If your application targets the full version of the .NET Framework 4 and cannot run on the .NET Framework 4 Client Profile, remove the client `<framework>` element by using a text editor and re-sign the manifest.

The following is a sample `<framework>` element that targets the .NET Framework 4 Client Profile:

XML

```
<framework targetVersion="4.0" profile="client" supportedRuntime="4.0.20506"
/>
```

Examples

The following example opens the user interface for Mage (*MageUI.exe*).

```
Console
```

```
mage
```

The following examples create a default deployment manifest and application manifest. These files are all created in the current working directory and are named `deploy.application` and `application.exe.manifest`, respectively.

```
Console
```

```
mage -New Deployment  
mage -New Application
```

The following example creates an application manifest populated with all of the assemblies and resource files from the current directory.

```
Console
```

```
mage -New Application -FromDirectory . -Version 1.0.0.0
```

The following example continues the previous example by specifying the deployment name and target microprocessor. It also specifies a URL against which ClickOnce should check for updates.

```
Console
```

```
mage -New Application -FromDirectory . -Name "Hello, World! Application" -  
Version 1.0.0.0 -Processor "x86" -ProviderUrl  
http://internalserver/HelloWorld/
```

The following example demonstrates how to create a pair of manifests for deploying a WPF application that will be hosted in a browser.

```
Console
```

```
mage -New Application -FromDirectory . -Version 1.0.0.0 -WPFBrowserApp true  
mage -New Deployment -AppManifest 1.0.0.0\application.manifest -  
WPFBrowserApp true
```

The following example creates an application manifest populated with all of the assemblies and resource files from the current directory and signs.

Console

```
mage -New Application -FromDirectory . -Version 1.0.0.0 -KeyContainer  
keypair.snk -CryptoProvider "Microsoft Enhanced Cryptographic Provider v1.0"
```

The following example updates a deployment manifest with information from an application manifest, and sets the code base for the location of the application manifest.

Console

```
mage -Update HelloWorld.deploy -AppManifest 1.0.0.0\application.manifest -  
AppCodeBase http://internalserver/HelloWorld.deploy
```

The following example edits the deployment manifest to force an update of the user's installed version.

Console

```
mage -Update c:\HelloWorldDeployment\HelloWorld.deploy -MinVersion 1.1.0.0
```

The following example tells the deployment manifest to retrieve the application manifest from another directory.

Console

```
mage -Update HelloWorld.deploy -AppCodeBase  
http://anotherserver/HelloWorld/1.1.0.0/
```

The following example signs an existing deployment manifest using a digital certificate in the current working directory.

Console

```
mage -Sign deploy.application -CertFile cert.pfx -Password <passwd>
```

The following example signs an existing deployment manifest using a digital certificate and private key in the current working directory.

Console

```
mage -Sign deploy.application -CertFile cert.pfx -KeyContainer keyfile.snk -  
CryptoProvider "Microsoft Enhanced Cryptographic Provider v1.0"
```

See also

- [ClickOnce Security and Deployment](#)
- [Walkthrough: Manually Deploying a ClickOnce Application](#)
- [Trusted Application Deployment Overview](#)
- [MageUI.exe \(Manifest Generation and Editing Tool, Graphical Client\)](#)
- [Developer command-line shells](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

MageUI.exe (Manifest Generation and Editing Tool, Graphical Client)

Article • 09/15/2021

MageUI.exe supports the same functionality as the command-line tool Mage.exe, but with a Windows-based user interface (UI). With this tool you can create, edit, and sign deployment and application manifests. New manifests that are created with MageUI.exe target the .NET Framework 4 Client Profile. Previous versions of MageUI.exe should be used to target previous .NET Framework versions. When adding or removing assemblies from a manifest, or re-signing existing manifests, MageUI.exe does not update the manifest to target .NET Framework 4 Client Profile. For more information, see [Mage.exe \(Manifest Generation and Editing Tool\)](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

Two versions of Mage.exe and MageUI.exe are included as a component of Visual Studio. To see version information, run MageUI.exe, select **Help**, and select **About**. This documentation describes version 4.0.x.x of Mage.exe and MageUI.exe.

ⓘ Note

MageUI.exe does not support the **compatibleFrameworks** element when saving an application manifest that has already been signed with a certificate using MageUI.exe. Instead, you must use [Mage.exe](#).

UIElement List

The following table lists the menu and toolbar items that are available.

Command	Menu	Shortcut	Description
Application Manifest	File, New		Creates a new application manifest.
Deployment Manifest	File, New		Creates a new deployment manifest.
Open	File	CTRL+O	Opens an existing deployment manifest, application manifest, or trust license for editing.

Command	Menu	Shortcut	Description
Close	File	CTRL+F4	Closes an open file. If you modify a file before closing it, MageUI.exe prompts you to re-sign the file with a public key, key pair, or stored certificate.
Save	File	CTRL+S	Saves to disk the document which currently has user input focus.
Save As	File		Saves a file to disk, enabling you to supply a new file name and/or location.
Save All	File		Saves the changes made to all files currently open within MageUI.exe.
Preferences	File		Opens the Preferences dialog box. See the following section for more information.
Exit	File	ALT+F4	Quits MageUI.exe.
Cut	Edit	CTRL+X	Removes the currently selected text from the application and moves it to the system Clipboard.
Copy	Edit	CTRL+C	Copies the currently selected text to the system Clipboard.
Paste	Edit	CTRL+V	Pastes text from the system Clipboard into the currently active text element.
Delete	Edit		Deletes an element currently selected in a list, such as a trust license on the Deployment Manifest tab.
Close All	Window		Closes all files currently open in MageUI.exe. If one or more files need saving, MageUI.exe prompts you to save them. MageUI.exe also prompts you to select a signing key for each unsigned or changed file.
About	Help		Displays version and copyright information about MageUI.exe.

Preferences Dialog Box

The **Preferences** dialog box contains the following elements.

UI Element	Description

UI Element	Description
Sign on save	Prompts you to sign a file whenever you save your modifications.
Use default signing certificate	Uses the key entered in the Certificate file text box to sign all files. This eliminates the signing prompt that typically appears when you save a file and Sign on Save is selected. Use the ellipsis (...) button next to the Certificate file text box to select a key file.
Digest algorithm	Specifies the algorithm to generate dependency digests with. Value must be "sha256RSA" or "sha1RSA". Uses SHA1 as the default. Used both in application and deployment manifests. If the user provides a certificate when saving the manifest, uses the algorithms in the certificate to generate dependency digests with.

Signing Options Dialog Box

The **Signing Options** dialog box appears when you save a manifest or trust license for the first time, or when you change a manifest or trust license. It only appears if the **Sign on Save** option in the **Preferences** dialog box is selected. You must be connected to the Internet when signing a manifest that specifies a value in the **TimeStamping URI** text box.

This dialog box contains the following elements.

UI Element	Description
Sign with certificate file	Signs the manifest with a digital certificate stored on the file system.
File	Provides an area to type the path to the .pfx file representing the certificate.
...	Opens a Choose File dialog box for selecting an existing .pfx file.
New	Generates a new .pfx that is not verifiable through a Certificate Authority (CA). For more information about the types of certificates used for signing ClickOnce deployments, see Trusted Application Deployment Overview .
Password	Provides an area to type the password used for signing with this certificate. If not applicable, can be left blank.
Sign with stored certificate	Displays a selectable list of digital certificates stored in your computer's certificate store.

UI Element	Description
TimeStamping URI	Displays the Uniform Resource Locator (URI) of a digital timestamping service. Timestamping the manifests prevents you from having to re-sign the manifests if your digital certificate expires before you deploy the next version of your application. For more information, see Windows root certificate program members and ClickOnce and Authenticode .
Don't Sign	Allows you to save the manifest without adding a signature from a digital certificate.

Tab and Panel Descriptions

When you open a document with MageUI.exe, it appears within its own tab page. Each tab contains a set of property panels. The panels contain grouped subsets of the document's data.

Application Manifest Tab

The **Application Manifest** tab displays the contents of an application manifest. The application manifest describes all files included with the deployment, and the permissions required for the application to run on the client.

The **Application Manifest** tab contains the following tabs.

UI Element	Description
Name	Specifies identifying information about this deployment.
Description	Specifies publisher, product, and support information.
Application Options	Specifies whether this is a browser application, and whether this manifest is the source of trust information.
Files	Specifies all of the files that constitute this deployment.
Permissions Required	Specifies the minimum permission set required by the application to run on a client.

Name Tab

The **Name** tab is displayed when you first create or open an application manifest. It uniquely identifies the deployment, and optionally specifies a valid target platform.

UI Element	Description
Name	Required. The name of the application manifest. Usually the same as the file name.
Version	Required. The version number of the deployment in the form <i>N.N.N.N</i> . Only the first major build number is required. For example, for version 1.0 of an application, valid values would include 1, 1.0, 1.0.0, and 1.0.0.0.
Processor	Optional. The machine architecture on which this deployment can run. The default is msil, or Microsoft Intermediate Language, which is the default format of all managed assemblies. Change this field if you have pre-compiled the assemblies in your application for a specific architecture. For more information about pre-compilation, see Ngen.exe (Native Image Generator) .
Culture	Optional. The two-part ISO country and region code in which this application runs. The default is neutral.
Public key token	Optional. The public key with which this application manifest has been signed. If this is a new or unsigned manifest, this field will appear as Unsigned.

Description Tab

This information is usually provided within the deployment manifest. These fields can only be modified if the **Use Application Manifest Trust Information** check box is selected on the **Application Options** tab.

UI Element	Description
Publisher	The name of the person or organization responsible for the application. This value is used as the Start menu folder name.
Product	The full product name. If you selected Install Locally for the Application Type element on the Deployment Options tab of the deployment manifest, this name will be what appears in the Start menu link and in Add or Remove Programs for this application.
Support Location	The URL from which customers can obtain help and support for the application.

Application Options Tab

UI Element	Description
------------	-------------

UI Element	Description
Windows Presentation Foundation Browser Application	Specifies whether this is a WPF application that runs in the browser as a XAML browser application (XBAP).
Use Application Manifest Trust Information	Specifies whether this manifest contains trust information.

Files Tab

UI Element	Description
Application directory	The directory in which the application's files reside. Use the ellipses (...) button to select the directory.
Populate	Adds all of the files in the application directory and subdirectories to the application manifest. If MageUI.exe finds a single executable file in the directory, it automatically marks this as the Entry Point, which is the file first executed when the ClickOnce application is launched on the client.
Application Files	Lists all of the files in the application. Each file has three editable attributes, discussed below.
File Type	<p>File Type can be one of four values:</p> <ul style="list-style-type: none"> - None. - Entry Point. The application's primary executable. Only one executable file can be marked as the entry point. - Data File. A file, such as an XML file, that supplies data to the application. - Icon File. An application icon, such as appears on the desktop or in the corner of an application's window.
Optional	Files marked optional are not downloaded on initial install or update, but may be downloaded at run time using the System.Deployment On-Demand API. For more information, see Walkthrough: Downloading Assemblies on Demand with the ClickOnce Deployment API Using the Designer .
Group	A label for a set of optional files. You can apply a Group label to a set of files, and use the On-Demand API to download a batch of files with a single API call.

Permissions Required Tab

Use the **Permissions Required** tab if you need to grant your application more access to the local computer than is granted by default. For more information, see [Securing ClickOnce Applications](#).

UI Element	Description
Permission set type	The minimum permission set required by this application to run on the client. For a description of these permission sets and which permissions they do or do not demand, see Named Permission Sets .
Details	The XML created for the application manifest to represent the permission set. Unless you have a good understanding of the application manifest XML format, you should not edit this XML manually. For more information, see ClickOnce Application Manifest .

Deployment Manifest Tab

The **Deployment Manifest** tab contains the following tabs.

UI Element	Description
Name	Specifies identifying information about this deployment.
Description	Specifies publisher, product, and support information.
Deployment Options	Specifies additional information about the deployment, such as the application type and the start location.
Update Options	Specifies how often ClickOnce should check for application updates.
Application Reference	Specifies the application manifest for this deployment.

Name Tab

The **Name** tab is displayed when you first create or open a deployment manifest. It uniquely identifies the deployment, and optionally specifies a valid target platform.

UI Element	Description
Name	Required. The name of the deployment manifest. Usually the same as the file name.
Version	Required. The version number of the deployment in the form <i>N.N.N.N</i> . Only the first major build number is required. For example, for version 1.0 of an application, valid values would include <code>1</code> , <code>1.0</code> , <code>1.0.0</code> , and <code>1.0.0.0</code> .

UI Element	Description
Processor	Optional. The machine architecture on which this deployment can run. The default is <code>msil</code> , or Microsoft Intermediate Language, the default format of all managed assemblies. Change this field if you have compiled the assemblies in your application for a specific architecture.
Culture	Optional. The two-part ISO country/region code in which this application runs. The default is <code>neutral</code> .
Public key token	Optional. The public key with which this deployment manifest has been signed. If this is a new or unsigned manifest, this field will appear as <code>Unsigned</code> .

Description Tab

UI Element	Description
Publisher	Required. The name of the person or organization responsible for the application. This value is used as the Start menu folder name.
Product	Required. The full product name. If you selected Install Locally for the Application Type element on the Deployment Options tab, this name will be what appears in the Start menu link and in Add or Remove Programs for this application.
Support Location	Optional. The URL from which customers can obtain help and support for the application.

Deployment Options Tab

UI Element	Description
Application Type	Optional. Specifies whether this application installs itself to the client computer (Install Locally), runs online (Online Only), or is a WPF application that runs in the browser (WPF Browser Application). The default is Install Locally .
Start Location	Optional. The URL from which the application should actually be started. Useful when deploying an application from a CD that should update itself from the Web.
Include Start Location (ProviderURL) in the manifest	Optional. Specifies the URL which ClickOnce will examine for application updates.

UI Element	Description
Automatically run application after installing	Required. Specifies that the ClickOnce application should run immediately after the initial installation from a URL. The default is the check box is selected.
Allow URL parameters to be passed to application	Required. Permits the transfer of parameter data to the ClickOnce application through a query string appended to the deployment manifest's URL. The default is the check box is cleared.
Use .deploy file extension	Required. When selected, all files in the application manifest must have the .deploy extension. The default is the check box is cleared.

Update Options Tab

The **Update Options** tab only contains options mentioned here when the **Application Type** selection box on the **Name** tab is set to **Install Locally**.

UI Element	Description
This application should check for updates	Specifies whether ClickOnce should check for application updates. If this check box is not selected, the application will not check for updates unless you update it programmatically by using the APIs in the System.Deployment.Application namespace.
Choose when the application should check for updates	Provides two options for update checks: <ul style="list-style-type: none"> - Before the application starts. The update check is performed prior to application execution. - After the application starts. The update check begins once the main form of the application has initialized, and will run the next time the application starts.
Update check frequency	Determines how often ClickOnce should check for updates: <ul style="list-style-type: none"> - Check every time the application runs. ClickOnce will perform an update check every time the user opens the application. - Check every: Select a time interval and a unit (hours, days, or weeks) that must elapse before checking for updates.
Specify a minimum required version for this application	Optional. Specifies that a specific version of your application is a required installation, preventing your users from working with an earlier version.

UI Element	Description
Version	Required if Specify a minimum required version for this application check box is selected. The version number supplied must be of the form <i>N.N.N.N</i> . Only the first major build number is required. For example, for version 1.0 of an application, valid values would include 1, 1.0, 1.0.0, and 1.0.0.0.

Application Reference Tab

The **Application Reference** tab contains the same fields as the **Name** tab described earlier in this topic. The one exception is the following field.

UI Element	Description
Select Manifest	Allows you to choose the application manifest. All of the other fields on this page will populate when you choose an application manifest.

See also

- [ClickOnce Security and Deployment](#)
- [Walkthrough: Manually Deploying a ClickOnce Application](#)
- [Mage.exe \(Manifest Generation and Editing Tool\)](#)

MDbg.exe (.NET Framework Command-Line Debugger)

Article • 09/15/2021

The .NET Framework Command-Line Debugger helps tools vendors and application developers find and fix bugs in programs that target the .NET Framework common language runtime. This tool uses the runtime debugging API to provide debugging services. You can use MDbg.exe to debug only managed code; there is no support for debugging unmanaged code.

This tool is available through NuGet. For installation information, see [MDbg 0.1.0](#). To run the tool, use the Package Manager Console. For more information about how to use the Package Manager Console, see the [Package Manager Console](#) article.

At the Package Manager prompt, type the following:

Syntax

Console

```
MDbg [ProgramName[arguments]] [options]
```

Commands

When you are in the debugger (as indicated by the `mdbg>` prompt), type one of the commands described in the next section:

command [*arguments*]

MDbg.exe commands are case-sensitive.

[] [Expand table](#)

Command	Description
<code>ap[rocess] [number]</code>	Switches to another debugged process or prints available processes. The numbers are not real process IDs (PIDs), but a 0-indexed list.
<code>a[ttach] [pid]</code>	Attaches to a process or prints available processes.
<code>b[reak] [ClassName.Method </code>	Sets a breakpoint at the specified method. Modules are scanned sequentially.

Command	Description
<code>FileName:LineNo]</code>	<ul style="list-style-type: none"> - break <i>FileName:LineNo</i> sets a breakpoint at a location in the source. - break <i>~number</i> sets a breakpoint on a symbol recently displayed with the x command. - break <i>module!ClassName.Method+Offset</i> sets a breakpoint on the fully qualified location.
<code>block[ingObjects]</code>	Displays monitor locks, which are blocking threads.
<code>ca[tch] [exceptionType]</code>	Causes the debugger to break on all exceptions, and not just on the unhandled exceptions.
<code>cl[earException]</code>	Marks the current exception as handled so that execution can continue. If the cause of the exception has not been dealt with, the exception may be quickly rethrown.
<code>conf[ig] [option value]</code>	<p>Displays all configurable options and shows how the options are invoked without any optional values. If the option is specified, sets <code>value</code> as the current option. The following options are currently available:</p> <ul style="list-style-type: none"> - <code>extpath</code> sets the path to search for extensions when the <code>load</code> command is used. - <code>extpath+</code> adds a path for loading extensions.
<code>del[ete]</code>	Deletes a breakpoint.
<code>de[tach]</code>	Detaches from a debugged process.
<code>d[own] [frames]</code>	Moves the active stack frame down.
<code>echo</code>	Echoes a message to the console.
<code>enableNotif[ication] <i>typeName</i> 0 1</code>	Enables (1) or disables (0) custom notifications for the specified type.
<code>ex[it] [exitcode]</code>	Exits the MDbg.exe shell, and optionally specifies the process exit code.
<code>fo[reach] [<i>OtherCommand</i>]</code>	Performs a command on all threads. <i>OtherCommand</i> is a valid command that operates on one thread; foreach <i>OtherCommand</i> performs the same command on all threads.
<code>f[unceval] [-ad <i>Num</i>] <i>functionName</i> [<i>args ...</i>]</code>	<p>Performs a function evaluation on the current active thread where the function to evaluate is <i>functionName</i>. The function name must be fully qualified, including namespaces.</p> <p>The <code>-ad</code> option specifies the application domain to use to resolve the function. If the <code>-ad</code> option is not specified, the application domain for resolution defaults to the application domain where the thread that is used for function evaluation is located.</p>

Command	Description
	If the function that is being evaluated is not static, the first parameter passed in should be a <code>this</code> pointer. All application domains are searched for arguments to the function evaluation..
	To request a value from an application domain, prefix the variable with the module and application domain name; for example, <code>funcEval -ad 0 System.Object.ToString hello.exe#0!MyClass.g_rootRef</code> . This command evaluates the function <code>System.Object.ToString</code> in the application domain <code>0</code> . Because the <code>ToString</code> method is an instance function, the first parameter must be a <code>this</code> pointer.
<code>g[o]</code>	Causes the program to continue until it encounters a breakpoint, the program exits, or an event (for example, an unhandled exception) causes the program to stop.
<code>h[elp] [command]</code>	Displays a description of all commands or a detailed description of a specified command.
-or-	
<code>? [command]</code>	
<code>ig[nore] [event]</code>	Causes the debugger to stop on unhandled exceptions only.
<code>int[ercept] <i>FrameNumber</i></code>	Rolls the debugger back to a specified frame number.
	If the debugger encounters an exception, use this command to roll the debugger back to the specified frame number. You can change the program state by using the <code>set</code> command and continue by using the <code>go</code> command.
<code>k[ill]</code>	Stops the active process.
<code>l[ist] [modules appdomains assemblies]</code>	Displays the loaded modules, application domains, or assemblies.
<code>lo[ad] <i>assemblyName</i></code>	Loads an extension in the following manner: The specified assembly is loaded and an attempt is then made to run the static method <code>LoadExtension</code> from the <code>Microsoft.Tools.Mdbg.Extension</code> type.
<code>log [eventType]</code>	Set or display the events to be logged.
<code>mo[de] [option on/off]</code>	Sets different debugger options. Use <code>mode</code> with no options to get a list of the debugging modes and their current settings.
<code>mon[itorInfo] <i>monitorReference</i></code>	Displays object monitor lock information.

Command	Description
new [o] <i>typeName</i> [<i>arguments...</i>]	Creates a new object of type <i>typeName</i> .
n [e x] t	Runs code and moves to the next line (even if the next line includes many function calls).
Opendump <i>pathToDumpFile</i>	Opens the specified dump file for debugging.
o [u t]	Moves to the end of the current function.
pa [t h] [<i>pathName</i>]	Searches the specified path for the source files if the location in the binaries is not available.
p [r int] [<i>var</i>] [-d]	Prints all variables in scope (print), prints the specified variable (print var), or prints the debugger variables (print -d).
print [e xception] [-r]	Prints the last exception on the current thread. Use the -r (recursive) option to traverse the InnerException property on the exception object to get information about the entire chain of exceptions.
pro [c essenum]	Displays the active processes.
q [u it] [<i>exitcode</i>]	Quits the MDbg.exe shell, optionally specifying the process exit code.
re [s ume] [* [~] <i>threadNumber</i>]	Resumes the current thread or the thread specified by the <i>threadNumber</i> parameter. If the <i>threadNumber</i> parameter is specified as * or if the thread number starts with ~, the command applies to all threads except the one specified by <i>threadNumber</i> .
	Resuming a non-suspended thread has no effect.
r [un] [-d(e bug)] -o(p timize) -enc [[<i>path_to_exe</i>] [<i>args_to_exe</i>]]	Stops the current process (if there is one) and starts a new one. If no executable argument is passed, this command runs the program that was previously executed with the run command. If the executable argument is provided, the specified program is run using the optionally supplied arguments. If class load, module load, and thread start events are ignored (as they are by default), the program stops on the first executable instruction of the main thread.
	You can force the debugger to just-in-time (JIT) compile the code by using one of the following three flags:
	- -d (e bug) disables optimizations. This is the default for MDbg.exe. - -o (p timize) forces the code to run more like it does outside the

Command	Description
	<p>debugger, but also makes the debugging experience more difficult. This is the default for use outside the debugger.</p> <ul style="list-style-type: none"> - <code>-enc</code> enables the Edit and Continue feature but incurs a performance hit.
<code>Set variable=value</code>	<p>Changes the value of any in-scope variable.</p> <p>You can also create your own debugger variables and assign reference values to them from within your application. These values act as handles to the original value, and even the original value is out of scope. All debugger variables must begin with <code>\$</code> (for example, <code>\$var</code>). Clear these handles by setting them to nothing using the following command:</p> <pre>set \$var=</pre>
<code>Setip [-il] number</code>	<p>Sets the current instruction pointer (IP) in the file to the specified position. If you specify the <code>-il</code> option, the number represents a common intermediate language (CIL) offset in the method. Otherwise, the number represents a source line number.</p>
<code>sh[ow] [lines]</code>	<p>Specifies the number of lines to show.</p>
<code>s[tep]</code>	<p>Moves execution into the next function on the current line, or moves to the next line if there is no function to step into.</p>
<code>su[pend] [* ~]threadNumber</code>	<p>Suspends the current thread or the thread specified by the <code>threadNumber</code> parameter. If <code>threadNumber</code> is specified as <code>*</code>, the command applies to all threads. If the thread number starts with <code>~</code>, the command applies to all threads except the one specified by <code>threadNumber</code>. Suspended threads are excluded from running when the process is run by either the <code>go</code> or <code>step</code> command. If there are no non-suspended threads in the process and you issue the <code>go</code> command, the process will not continue. In that case, press CTRL-C to break into the process.</p>
<code>sy[mbol] commandName [commandValue]</code>	<p>Specifies one of the following commands:</p> <ul style="list-style-type: none"> - <code>symbol path ["value"]</code> - Displays or sets the current symbol path. - <code>symbol addpath "value"</code> - Adds to your current symbol path. - <code>symbol reload ["module"]</code> - Reloads either all symbols or the symbols for the specified module. - <code>symbol list [module]</code> - Shows the currently loaded symbols for either all modules or the specified module.
<code>thread] [newThread] [-nick nickname]</code>	<p>The <code>thread</code> command with no parameters displays all managed threads in the current process. Threads are usually identified by their thread numbers; however, if the thread has an assigned nickname, the nickname is displayed instead. You can use the <code>-nick</code> parameter to assign a nickname to a thread.</p>

Command	Description
	<p>- thread -nick <i>threadName</i> assigns a nickname to the currently running thread.</p> <p>Nicknames cannot be numbers. If the current thread already has an assigned nickname, the old nickname is replaced with the new one. If the new nickname is an empty string (""), the nickname for the current thread is deleted and no new nickname is assigned to the thread.</p>
u[p]	Moves the active stack frame up.
uwgc[handle] [var] [address]	Prints the variable tracked by a handle. The handle can be specified by name or address.
when	Displays the currently active <code>when</code> statements.
	<p>when delete all num [num [num ...]] - Deletes the <code>when</code> statement specified by the number, or all <code>when</code> statements if <code>all</code> is specified.</p> <p>when stopReason [specific_condition] do cmd [cmd [cmd ...]] - The <code>stopReason</code> parameter can be one of the following:</p> <pre data-bbox="505 1058 1364 1272">StepComplete, ProcessExited, ThreadCreated, BreakpointHit, ModuleLoaded, ClassLoaded, AssemblyLoaded, AssemblyUnloaded, ControlCTrapped, ExceptionThrown, UnhandledExceptionThrown, AsyncStop, AttachComplete, UserBreak, EvalComplete, EvalException, RemapOpportunityReached, NativeStop.</pre> <p><code>specific_condition</code> can be one of the following:</p> <ul data-bbox="505 1407 1368 1654" style="list-style-type: none"> - <i>number</i> - For <code>ThreadCreated</code> and <code>BreakpointHit</code>, triggers action only when stopped by a thread ID/breakpoint number with same value. - <code>[!]name</code> - For <code>ModuleLoaded</code>, <code>ClassLoaded</code>, <code>AssemblyLoaded</code>, <code>AssemblyUnloaded</code>, <code>ExceptionThrown</code>, and <code>UnhandledExceptionThrown</code>, triggers action only when the name matches the name of the <code>stopReason</code>. <p><code>specific_condition</code> must be empty for other values of <code>stopReason</code>.</p>
w[here] [-v] [-c depth] [threadID]	Displays debug information about stack frames.
	<ul data-bbox="505 1868 1384 2070" style="list-style-type: none"> - The <code>-v</code> option provides verbose information about each displayed stack frame. - Specifying a number for <code>depth</code> limits how many frames are displayed. Use the <code>all</code> command to display all frames. The default is 100. - If you specify the <code>threadID</code> parameter, you can control which thread is

Command	Description
	associated with the stack. The default is the current thread only. Use the all command to get all threads.
x [-c <i>numSymbols</i>] [module[! <i>pattern</i>]]	<p>Displays functions that match the pattern for a module.</p> <p>If <i>numSymbols</i> is specified, the output is limited to the specified number. If ! (indicating a regular expression) is not specified for <i>pattern</i>, all functions are displayed. If <i>module</i> is not provided, all loaded modules are displayed. Symbols (~#) can be used to set breakpoints using the break command.</p>

Remarks

Compile the application to be debugged by using compiler-specific flags that cause your compiler to generate debugging symbols. Refer to your compiler's documentation for more information about these flags. You can debug optimized applications, but some debugging information will be missing. For example, many local variables will not be visible and source lines will be inaccurate.

After you compile your application, type **mdbg** at the command prompt to start a debugging session, as shown in the following example.

```
Console

C:\Program Files\Microsoft Visual Studio 8\VC>mdbg
MDbg (Managed debugger) v2.0.50727.42 (RTM.050727-4200) started.
Copyright (C) Microsoft Corporation. All rights reserved.

For information about commands type "help";
to exit program type "quit".
mdbg>
```

The **mdbg>** prompt indicates that you are in the debugger.

Once you are in the debugger, use the commands and arguments described in the previous section.

See also

- [Tools](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Mgmtclassgen.exe (Management Strongly Typed Class Generator)

Article • 09/15/2021

The Management Strongly Typed Class Generator tool enables you to quickly generate an early-bound managed class for a specified Windows Management Instrumentation (WMI) class. The generated class simplifies the code you must write to access an instance of the WMI class.

Syntax

Console

```
mgmtclassgen  
WMIClass [options]
```

Argument	Description
<i>WMIClass</i>	The Windows Management Instrumentation class for which to generate an early-bound managed class.

Option	Description
<i>/l language</i>	Specifies the language in which to generate the early-bound managed class. You can specify CS (C#; default), VB (Visual Basic), MC (C++), or JS (JScript) as the language argument.
<i>/m machine</i>	Specifies the computer to connect to, where the WMI class resides. The default is the local computer.
<i>/n path</i>	Specifies the path to the WMI namespace that contains the WMI class. If you do not specify this option, the tool generates code for <i>WMIClass</i> in the default Root\cimv2 namespace.
<i>/o classnamespace</i>	Specifies the .NET namespace in which to generate the managed code class. If you do not specify this option, the tool generates the namespace using the WMI namespace and the schema prefix. The schema prefix is the part of the class name preceding the underscore character. For example, for the Win32_OperatingSystem class in the Root\cimv2 namespace, the tool would generate the class in ROOT.CIMV2.Win32 .

Option	Description
/p <i>filepath</i>	Specifies the path to the file in which to save the generated code. If you do not specify this option, the tool creates the file in the current directory. It names the class and file in which it generates the class using the <i>WMIClass</i> argument. The name of the class and the file are the same as the name of the <i>WMIClass</i> . If <i>WMIClass</i> contains an underscore character, the tool uses the part of the class name following the underscore character. For example, if the <i>WMIClass</i> name is in the format Win32_LogicalDisk , the generated class and file is named "logicaldisk". If a file already exists, the tool overwrites the existing file.
/pw <i>password</i>	Specifies the password to use when logging on to a computer specified by the /m option.
/u <i>user name</i>	Specifies the user name to use when logging on to a computer specified by the /m option.
/?	Displays command syntax and options for the tool.

Remarks

Mgmtclassgen.exe uses the [ManagementClass.GetStronglyTypedClassCode](#) method. Therefore, you can use any custom code provider to generate code in managed languages other than C#, Visual Basic, and JScript.

Note that generated classes are bound to the schema for which they are generated. If the underlying schema changes, you must regenerate the class if you want it to reflect changes to the schema.

The following table shows how WMI Common Information Model (CIM) types map to data types in a generated class:

CIM type	Data type in the generated class
CIM_SINT8	SByte
CIM_UINT8	Byte
CIM_SINT16	Int16
CIM_UINT16	UInt16
CIM_SINT32	Int32
CIM_UINT32	UInt32
CIM_SINT64	Int64

CIM type	Data type in the generated class
CIM_UINT64	UInt64
CIM_REAL32	Single
CIM_REAL64	Double
CIM_BOOLEAN	Boolean
CIM_String	String
CIM_DATETIME	DateTime or TimeSpan
CIM_REFERENCE	ManagementPath
CIM_CHAR16	Char
CIM_OBJECT	ManagementBaseObject
CIM_IUNKNOWN	Object
CIM_ARRAY	Array of the above mentioned objects

Note the following behaviors when you generate a WMI class:

- It is possible for a standard public property or method to have the same name as an existing property or method. If this occurs, the tool changes the name of the property or method in the generated class to avoid naming conflicts.
- It is possible for the name of a property or method in a generated class to be a keyword in the target programming language. If this occurs, the tool changes the name of the property or method in the generated class to avoid naming conflicts.
- In WMI, qualifiers are modifiers that contain information to describe a class, instance, property, or method. WMI uses standard qualifiers such as **Read**, **Write**, and **Key** to describe a property in a generated class. For example, a property that is modified with a **Read** qualifier is defined only with a property **get** accessor in the generated class. Because a property marked with the **Read** qualifier is intended to be read-only, a **set** accessor is not defined.
- A numeric property can be modified by the **Values** and **ValueMaps** qualifiers to indicate that the property can be set only to specified permissible values. An enumeration is generated with these **Values** and **ValueMaps** and the property is mapped to the enumeration.
- The WMI uses the term singleton to describe a class that can have only one instance. Therefore, the parameterless constructor for a singleton class will initialize

the class to the only instance of the class.

- A WMI class can have properties that are objects. When you generate a strongly typed class for this type of WMI class, you should consider generating strongly typed classes for the types of the embedded object properties. This will allow you to access the embedded objects in a strongly typed manner. Note that the generated code might not be able to detect the type of the embedded object. In this case, a comment will be created in the generated code to notify you of this issue. You can then modify the generated code to type the property to the other generated class.
- In WMI, the data value of the CIM_DATETIME data type can represent either a specific date and time or a time interval. If the data value represents a date and time, the data type in the generated class is **DateTime**. If the data value represents a time interval, the data type in the generated class is **TimeSpan**.

You can alternately generate a strongly typed class using the Server Explorer Management Extension in Visual Studio .NET.

For more information about WMI, see the [Windows Management Instrumentation](#) topic in the Platform SDK documentation.

Examples

The following command generates a managed class in C# code for the **Win32_LogicalDisk** WMI class in the **Root\cimv2** namespace. The tool writes the managed class to the source file at **c:\disk.cs** in the **ROOT.CIMV2.Win32** namespace.

Console

```
mgmtclassgen Win32_LogicalDisk /n root\cimv2 /1 CS /p c:\disk.cs
```

The following code example shows how to use a generated class programmatically. First, an instance of the class is enumerated and the path is printed. Next, an instance of the generated class to be initialized is created with an instance of WMI. **Process** is the class generated for **Win32_Process** and **LogicalDisk** is the class generated for **Win32_LogicalDisk** in the **Root\cimv2** namespace.

C#

```
using System;
using System.Management;
using ROOT.CIMV2.Win32;
```

```
public class App
{
    public static void Main()
    {
        // Enumerate instances of the Win32_process.
        // Print the Name property of the instance.
        foreach(Process ps in Process.GetInstances())
        {
            Console.WriteLine(ps.Name);
        }

        // Initialize the instance of LogicalDisk with
        // the WMI instance pointing to logical drive d:.
        LogicalDisk dskD = new LogicalDisk(new ManagementPath(
            "win32_LogicalDisk.DeviceId=\"d:\\""));
        Console.WriteLine(dskD.Caption);
    }
}
```

See also

- [System.Management](#)
- [ManagementClass.GetStronglyTypedClassCode](#)
- [System.CodeDom.Compiler.CodeDomProvider](#)
- [Tools](#)
- [Developer command-line shells](#)

Mpgo.exe (Managed Profile Guided Optimization Tool)

Article • 09/15/2021

The Managed Profile Guided Optimization Tool (Mpgo.exe) is a command-line tool that uses common end-user scenarios to optimize the native image assemblies that are created by the [Native Image Generator \(Ngen.exe\)](#). This tool enables you to run training scenarios that generate profile data. The [Native Image Generator \(Ngen.exe\)](#) uses this data to optimize its generated native image application assemblies. A training scenario is a trial run of an expected use of your application. Mpgo.exe is available in Visual Studio Ultimate 2012 and later versions. Starting with Visual Studio 2013, you can also use Mpgo.exe to optimize Windows 8.x Store apps.

Profile-guided optimization improves application startup time, memory utilization (working set size), and throughput by gathering data from training scenarios and using it to optimize the layout of native images.

When you encounter performance issues with startup time and working set size for Intermediate Language (IL) assemblies, we recommend that you first use Ngen.exe to eliminate just-in-time (JIT) compilation costs and to facilitate code sharing. If you need additional improvements, you can then use Mpgo.exe to further optimize your application. You can use the performance data from the un-optimized native image assemblies as a baseline to evaluate the performance gains. Using Mpgo.exe may result in faster cold startup times and a smaller working set size. Mpgo.exe adds information to IL assemblies that Ngen.exe uses to create optimized native image assemblies. For more information, see the entry [Improving Launch Performance for your Desktop Applications](#) in the .NET blog.

This tool is automatically installed with Visual Studio. To run the tool, use [Developer Command Prompt or Developer PowerShell](#) with administrator credentials.

Enter the following command at the command prompt:

For desktop apps:

Console

```
mpgo -Scenario <command> [-Import <directory>] -AssemblyList <assembly1>
<assembly2> ... -OutDir <directory> [options]
```

For Windows 8.x Store apps:

Console

```
mpgo -Scenario <packageName> -AppID <appId> -Timeout <seconds>
```

Parameters

All arguments to Mpgo.exe are case-insensitive. Commands are prefixed with a dash.

Note

You can use either `-Scenario` or `-Import` as a required command, but not both.

None of the required parameters are used if you specify the `-Reset` option.

Required parameter	Description

Required parameter	Description
<p>-Scenario <i><command></i></p> <p>—or—</p> <p>-Scenario <i><packageName></i></p> <p>-or-</p>	<p>For desktop apps, use <code>-Scenario</code> to specify the command to run the application you want to optimize, including any command-line arguments. Use three sets of double quotation marks around <i>command</i> if it specifies a path that includes spaces; for example: <code>mpgo.exe -scenario """C:\My App\myapp.exe""" -assemblylist """C:\My App\myapp.exe"""" -outdir "C:\optimized files"</code>. Do not use double quotation marks; they will not work correctly if <i>command</i> includes spaces.</p>
<p>-Import <i><directory></i></p>	<p>For Windows 8.x Store apps, use <code>-Scenario</code> to specify the package that you want to generate profile information for. If you specify the package display name or the package family name instead of the full package name, Mpgo.exe will select the package that matches the name you provided if there is only one match. If multiple packages match the specified name, Mpgo.exe will prompt you to choose a package.</p>
	<p>—or—</p> <p>Use <code>-Import</code> to specify that optimization data from previously optimized assemblies should be used to optimize the assemblies in <code>-AssemblyList</code>. <i>directory</i> specifies the directory that contains the previously optimized files. The assemblies specified in <code>-AssemblyList</code> or <code>-AssemblyListFile</code> are the new versions of the assemblies to be optimized using the data from the imported files. Using optimization data from older version of assemblies enables you to optimize newer versions of assemblies without re-running the scenario. However, if the imported and target assemblies include significantly different code, the optimization data will be ineffective. The assembly names specified in <code>-AssemblyList</code> or <code>-AssemblyListFile</code> must be present in the directory specified by <code>-Import directory</code>. Use three sets of double quotation marks around <i>directory</i> if it specifies a path that includes spaces.</p>
	<p>You must specify either <code>-Scenario</code> or <code>-Import</code>, but not both parameters.</p>
<p>-OutDir <i><directory></i></p>	<p>The directory in which to place the optimized assemblies. If an assembly already exists in the output directory folder, a new copy is created and an index number is appended to its name; for example: <i>assemblyname-1.exe</i>. Use double quotation marks around <i>directory</i> if it specifies a path that contains spaces.</p>

Required parameter	Description
-AssemblyList <i><assembly1></i> <i>assembly2 ...></i> —or— —or— —	A list of assemblies (including .exe and .dll files), separated by spaces, that you want collect profile information about. You can specify <code>C:\Dir*.dll</code> or <code>*.dll</code> to select all the assemblies in the designated or current working directory. See the Remarks section for more information.
AssemblyListFile <i><file></i>	A text file that contains the list of assemblies you want to collect profile information about, listed one assembly per line. If an assembly name begins with a hyphen (-), use an assembly file list or rename the assembly.
-AppID <i><appId></i>	The ID of the application in the specified package. If you use the wildcard (*), Mpgo.exe will try to enumerate the AppIDs in the package and will fall back to <i><package_family_name>!App</i> if it fails. If you specify a string that is prefixed by an exclamation point (!), Mpgo.exe will concatenate the package family name with the argument provided.
-Timeout <i><seconds></i>	The amount of time to allow the Windows 8.x Store app to run before the app exits.

Optional parameter	Description
-64bit	Instruments the assemblies for 64-bit systems. You must specify this parameter for 64-bit assemblies, even if your assembly declares itself as 64 bit.
-ExeConfig <i><filename></i>	Specifies the configuration file that your scenario uses to provide version and loader information.
-f	Forces the inclusion of the profile data in a binary assembly, even if it's signed. If the assembly is signed, it must be re-signed; otherwise, the assembly will fail to load and run.
-Reset	Resets the environment to make certain that an aborted profiling session doesn't affect your assemblies, and then quits. The environment is reset by default before and after a profiling session.
-Timeout <i><time in seconds></i>	Specifies the profiling duration in seconds. Use a value that is slightly more than your observed startup times for GUI applications. At the end of the time-out period, the profile data is recorded although the application continues to run. If you don't set this option, profiling will continue until application shutdown, at which time the data will be recorded.

Optional parameter	Description
- <code>LeaveNativeImages</code>	Specifies that the instrumented native images shouldn't be removed after running the scenario. This option is primarily used when you're getting the application that you specified for the scenario running. It will prevent the recreation of native images for subsequent runs of Mpg.exe. When you have finished running your application, there may be orphaned native images in the cache if you specify this option. In this case, run Mpg.exe with the same scenario and assembly list and use the <code>-RemoveNativeImages</code> parameter to remove these native images.
- <code>RemoveNativeImages</code>	Cleans up from a run where <code>-LeaveNativeImages</code> was specified. If you specify <code>-RemoveNativeImages</code> , Mpg.exe ignores any arguments except <code>-64bit</code> and <code>-AssemblyList</code> , and exits after removing all instrumented native images.

Remarks

You can use both `-AssemblyList` and `-AssemblyListFile` multiple times on the command line.

If you do not specify full path names when specifying assemblies, Mpg.exe looks in the current directory. If you specify an incorrect path, Mpg.exe displays an error message but continues to generate data for other assemblies. If you specify an assembly that is not loaded during the training scenario, no training data is generated for that assembly.

If an assembly in the list is in the global assembly cache, it will not be updated to contain the profile information. Remove it from the global assembly cache to collect profile information.

The use of Ngen.exe and Mpg.exe is recommended only for large managed applications, because the benefit of precompiled native images is typically seen only when it eliminates significant JIT compilation at run time. Running Mpg.exe on "Hello World" style applications that aren't working-set intensive will not provide any benefits, and Mpg.exe may even fail to gather profile data.

ⓘ Note

Ngen.exe and Mpg.exe are not recommended for ASP.NET applications and Windows Communication Foundation (WCF) services.

To Use Mpg.exe

1. Use a computer that has the Visual Studio Ultimate 2012 and your application installed.
2. Run Mpg.exe as an administrator with the necessary parameters. See the next section for sample commands.

The optimized intermediate language (IL) assemblies are created in the folder specified by the `-OutDir` parameter (in the examples, this is the `C:\Optimized` folder).
3. Replace the IL assemblies you used for Ngen.exe with the new IL assemblies that contain the profile information from the directory specified by `-OutDir`.
4. The application setup (using the images provided by Mpg.exe) will install optimized native images.

Suggested Workflow

1. Create a set of optimized IL assemblies by using Mpg.exe with the `-Scenario` parameter.
2. Check the optimized IL assemblies into source control.
3. In the build process, call Mpg.exe with the `-Import` parameter as a post-build step to generate optimized IL images to pass to Ngen.exe.

This process ensures that all assemblies have optimization data. If you check in updated optimized assemblies (steps 1 and 2) more frequently, the performance numbers will be more consistent throughout product development.

Using Mpg.exe from Visual Studio

You can run Mpg.exe from Visual Studio (see the article [How to: Specify Build Events \(C#\)](#)) with the following restrictions:

- You cannot use quoted paths with trailing slash marks, because Visual Studio macros also use trailing slash marks by default. (For example, `-OutDir "C:\Output Folder\"` is invalid.) To work around this restriction, you can escape the trailing slash. (For example, use `-OutDir "$(OutDir)\\"` instead.)

- By default, Mpgo.exe is not on the Visual Studio build path. You must either add the path to Visual Studio or specify the full path on the Mpgo command line. You can use either the `-Scenario` or the `-Import` parameter in the post-build event in Visual Studio. However, the typical process is to use `-Scenario` one time from a Developer Command Prompt for Visual Studio, and then use `-Import` to update the optimized assemblies after each build; for example:

```
C:\Program Files\Microsoft Visual Studio 11.0\Team Tools\Performance Tools\mpgo.exe" -import "$(OutDir)tmp" -assemblylist "$(TargetPath)" -outdir "$(OutDir)\".
```

Examples

The following Mpgo.exe command from a Developer Command Prompt for Visual Studio optimizes a tax application:

Console

```
mpgo -scenario "C:\MyApp\MyTax.exe /params par" -AssemblyList Mytax.dll
MyTaxUtil2011.dll -OutDir C:\Optimized -TimeOut 15
```

The following Mpgo.exe command optimizes a sound application:

Console

```
mpgo -scenario "C:\MyApp\wav2wma.exe -input song1.wav -output song1.wma" -
AssemblyList transcode.dll -OutDir C:\Optimized -TimeOut 15
```

The following Mpgo.exe command uses data from previously optimized assemblies to optimize newer versions of the assemblies:

Console

```
mpgo.exe -import "C:\Optimized" -assemblylist "C:\MyApp\MyTax.dll"
"C:\MyApp\MyTaxUtil2011.dll" -outdir C:\ReOptimized
```

See also

- [Ngen.exe \(Native Image Generator\)](#)
- [Developer command-line shells](#)
- [Improving Launch Performance for your Desktop Applications](#)
- [An Overview of Performance Improvements in .NET Framework 4.5](#)

Ngen.exe (Native Image Generator)

Article • 10/26/2022

The Native Image Generator (Ngen.exe) is a tool that improves the performance of managed applications. Ngen.exe creates native images, which are files containing compiled processor-specific machine code, and installs them into the native image cache on the local computer. The runtime can use native images from the cache instead of using the just-in-time (JIT) compiler to compile the original assembly.

ⓘ Note

Ngen.exe compiles native images for assemblies that target .NET Framework only. The equivalent native image generator for .NET Core is [CrossGen](#).

Changes to Ngen.exe in .NET Framework 4:

- Ngen.exe now compiles assemblies with full trust, and code access security (CAS) policy is no longer evaluated.
- Native images that are generated with Ngen.exe can no longer be loaded into applications that are running in partial trust.

Changes to Ngen.exe in the .NET Framework version 2.0:

- Installing an assembly also installs its dependencies, simplifying the syntax of Ngen.exe.
- Native images can now be shared across application domains.
- A new action, `update`, re-creates images that have been invalidated.
- Actions can be deferred for execution by a service that uses idle time on the computer to generate and install images.
- Some causes of image invalidation have been eliminated.

On Windows 8, see [Native Image Task](#).

For additional information on using Ngen.exe and the native image service, see [Native Image Service](#).

ⓘ Note

Ngen.exe syntax for versions 1.0 and 1.1 of the .NET Framework can be found in [Native Image Generator \(Ngen.exe\) Legacy Syntax](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
ngen action [options]
```

Console

```
ngen /? | /help
```

Actions

The following table shows the syntax of each `action`. For descriptions of the individual parts of an `action`, see the [Arguments](#), [Priority Levels](#), [Scenarios](#), and [Config](#) tables. The [Options](#) table describes the `options` and the help switches.

[+] [Expand table](#)

Action	Description
<code>install [assemblyName assemblyPath] [scenarios] [config] [/queue [: {1 2 3}]]</code>	Generate native images for an assembly and its dependencies and install the images in the native image cache. If <code>/queue</code> is specified, the action is queued for the native image service. The default priority is 3. See the Priority Levels table.
<code>uninstall [assemblyName assemblyPath] [scenarios] [config]</code>	Delete the native images of an assembly and its dependencies from the native image cache. To uninstall a single image and its dependencies, use the same command-line arguments that were used to install the

Action	Description
	<p>image. Note: Starting with .NET Framework 4, the action <code>uninstall</code> * is no longer supported.</p>
<code>update [/queue]</code>	<p>Update native images that have become invalid.</p> <p>If <code>/queue</code> is specified, the updates are queued for the native image service. Updates are always scheduled at priority 3, so they run when the computer is idle.</p>
<code>display [assemblyName assemblyPath]</code>	<p>Display the state of the native images for an assembly and its dependencies.</p> <p>If no argument is supplied, everything in the native image cache is displayed.</p>
<code>executeQueuedItems [1 2 3]</code> -or- <code>eqi [1 2 3]</code>	<p>Execute queued compilation jobs.</p> <p>If a priority is specified, compilation jobs with greater or equal priority are executed. If no priority is specified, all queued compilation jobs are executed.</p>
<code>queue {pause continue status}</code>	Pause the native image service, allow the paused service to continue, or query the status of the service.

Arguments

[\[\] Expand table](#)

Argument	Description
<code>assemblyName</code>	<p>The full display name of the assembly. For example, <code>"myAssembly, Version=2.0.0.0, Culture=neutral, PublicKeyToken=0038abc9deabfle5"</code>. Note: You can supply a partial assembly name, such as <code>myAssembly</code>, for the <code>display</code> and <code>uninstall</code> actions.</p> <p>Only one assembly can be specified per Ngen.exe command line.</p>
<code>assemblyPath</code>	<p>The explicit path of the assembly. You can specify a full or relative path.</p> <p>If you specify a file name without a path, the assembly must be located in the current directory.</p> <p>Only one assembly can be specified per Ngen.exe command line.</p>

Priority Levels

[\[\] Expand table](#)

Priority	Description
1	Native images are generated and installed immediately, without waiting for idle time.
2	Native images are generated and installed without waiting for idle time, but after all priority 1 actions (and their dependencies) have completed.
3	Native images are installed when the native image service detects that the computer is idle. See Native Image Service .

Scenarios

[\[\] Expand table](#)

Scenario	Description
/Debug	Generate native images that can be used under a debugger.
/Profile	Generate native images that can be used under a profiler.
/NoDependencies	Generate the minimum number of native images required by the specified scenario options.

Config

[\[\] Expand table](#)

Configuration	Description
/ExeConfig: exePath	Use the configuration of the specified executable assembly. Ngen.exe needs to make the same decisions as the loader when binding to dependencies. When a shared component is loaded at run time, using the Load method, the application's configuration file determines the dependencies that are loaded for the shared component — for example, the version of a dependency that is loaded. The <code>/ExeConfig</code> switch gives Ngen.exe guidance on which dependencies would be loaded at run time.
/AppBase: directoryPath	When locating dependencies, use the specified directory as the application base.

Options

Option	Description
/nologo	Suppress the Microsoft startup banner display.
/silent	Suppress the display of success messages.
/verbose	Display detailed information for debugging.
/help, /?	Display command syntax and options for the current release.

Remarks

To run Ngen.exe, you must have administrative privileges.

⊗ Caution

Do not run Ngen.exe on assemblies that are not fully trusted. Starting with .NET Framework 4, Ngen.exe compiles assemblies with full trust, and code access security (CAS) policy is no longer evaluated.

Starting with .NET Framework 4, the native images that are generated with Ngen.exe can no longer be loaded into applications that are running in partial trust. Instead, the just-in-time (JIT) compiler is invoked.

Ngen.exe generates native images for the assembly specified by the `assemblyname` argument to the `install` action and all its dependencies. Dependencies are determined from references in the assembly manifest. The only scenario in which you need to install a dependency separately is when the application loads it using reflection, for example by calling the [Assembly.Load](#) method.

ⓘ Important

Do not use the [Assembly.LoadFrom](#) method with native images. An image loaded with this method cannot be used by other assemblies in the execution context.

Ngen.exe maintains a count on dependencies. For example, suppose `MyAssembly.exe` and `YourAssembly.exe` are both installed in the native image cache, and both have references to `OurDependency.dll`. If `MyAssembly.exe` is uninstalled, `OurDependency.dll` is not uninstalled. It is only removed when `YourAssembly.exe` is also uninstalled.

If you are generating a native image for an assembly in the global assembly cache, specify its display name. See [Assembly.FullName](#).

The native images that Ngen.exe generates can be shared across application domains. This means you can use Ngen.exe in application scenarios that require assemblies to be shared across application domains. To specify domain neutrality:

- Apply the [LoaderOptimizationAttribute](#) attribute to your application.
- Set the [AppDomainSetup.LoaderOptimization](#) property when you create setup information for a new application domain.

Always use domain-neutral code when loading the same assembly into multiple application domains. If a native image is loaded into a nonshared application domain after having been loaded into a shared domain, it cannot be used.

 **Note**

Domain-neutral code cannot be unloaded, and performance may be slightly slower, particularly when accessing static members.

In this Remarks section:

- [Generating images for different scenarios](#)
 - [Improved memory use](#)
 - [Faster application startup](#)
 - [Summary of usage considerations](#)
- [Importance of assembly base addresses](#)
- [Hard binding](#)
 - [Specifying a binding hint for a dependency](#)
 - [Specifying a default binding hint for an assembly](#)
- [Deferred processing](#)
- [Native images and JIT compilation](#)
 - [Invalid images](#)

- Troubleshooting
 - [Assembly Binding Log Viewer](#)
 - [The JITCompilationStart managed debugging assistant](#)
 - [Opting out of native image generation](#)

Generating images for different scenarios

After you have generated a native image for an assembly, the runtime automatically attempts to locate and use this native image each time it runs the assembly. Multiple images can be generated, depending on usage scenarios.

For example, if you run an assembly in a debugging or profiling scenario, the runtime looks for a native image that was generated with the `/Debug` or `/Profile` options. If it is unable to find a matching native image, the runtime reverts to standard JIT compilation. The only way to debug native images is to create a native image with the `/Debug` option.

The `uninstall` action also recognizes scenarios, so you can uninstall all scenarios or only selected scenarios.

Determining when to Use native images

Native images can provide performance improvements in two areas: improved memory use and reduced startup time.

Note

Performance of native images depends on a number of factors that make analysis difficult, such as code and data access patterns, how many calls are made across module boundaries, and how many dependencies have already been loaded by other applications. The only way to determine whether native images benefit your application is by careful performance measurements in your key deployment scenarios.

Improved memory use

Native images can significantly improve memory use when code is shared between processes. Native images are Windows PE files, so a single copy of a .dll file can be

shared by multiple processes; by contrast, native code produced by the JIT compiler is stored in private memory and cannot be shared.

Applications that are run under terminal services can also benefit from shared code pages.

In addition, not loading the JIT compiler saves a fixed amount of memory for each application instance.

Faster application startup

Precompiling assemblies with Ngen.exe can improve the startup time for some applications. In general, gains can be made when applications share component assemblies because after the first application has been started the shared components are already loaded for subsequent applications. Cold startup, in which all the assemblies in an application must be loaded from the hard disk, does not benefit as much from native images because the hard disk access time predominates.

Hard binding can affect startup time, because all images that are hard bound to the main application assembly must be loaded at the same time.

Note

Before the .NET Framework 3.5 Service Pack 1, you should put shared, strong-named components in the global assembly cache, because the loader performs extra validation on strong-named assemblies that are not in the global assembly cache, effectively eliminating any improvement in startup time gained by using native images. Optimizations that were introduced in the .NET Framework 3.5 SP1 removed the extra validation.

Summary of usage considerations

The following general considerations and application considerations may assist you in deciding whether to undertake the effort of evaluating native images for your application:

- Native images load faster than CIL because they eliminate the need for many startup activities, such as JIT compilation and type-safety verification.
- Native images require a smaller initial working set because there is no need for the JIT compiler.

- Native images enable code sharing between processes.
- Native images require more hard disk space than CIL assemblies and may require considerable time to generate.
- Native images must be maintained.
 - Images need to be regenerated when the original assembly or one of its dependencies is serviced.
 - A single assembly may need multiple native images for use in different applications or different scenarios. For example, the configuration information in two applications might result in different binding decisions for the same dependent assembly.
 - Native images must be generated by an administrator; that is, from a Windows account in the Administrators group.

In addition to these general considerations, the nature of your application must be considered when determining whether native images might provide a performance benefit:

- If your application runs in an environment that uses many shared components, native images allow the components to be shared by multiple processes.
- If your application uses multiple application domains, native images allow code pages to be shared across domains.

 **Note**

In the .NET Framework versions 1.0 and 1.1, native images cannot be shared across application domains. This is not the case in version 2.0 or later.

- If your application will be run under Terminal Server, native images allow sharing of code pages.
- Large applications generally benefit from compilation to native images. Small applications generally do not benefit.
- For long-running applications, run-time JIT compilation performs slightly better than native images. (Hard binding can mitigate this performance difference to some degree.)

Importance of assembly base addresses

Because native images are Windows PE files, they are subject to the same rebasing issues as other executable files. The performance cost of relocation is even more pronounced if hard binding is employed.

To set the base address for a native image, use the appropriate option of your compiler to set the base address for the assembly. Ngen.exe uses this base address for the native image.

ⓘ Note

Native images are larger than the managed assemblies from which they were created. Base addresses must be calculated to allow for these larger sizes.

You can use a tool such as dumpbin.exe to view the preferred base address of a native image.

Hard binding

Hard binding increases throughput and reduces working set size for native images. The disadvantage of hard binding is that all the images that are hard bound to an assembly must be loaded when the assembly is loaded. This can significantly increase startup time for a large application.

Hard binding is appropriate for dependencies that are loaded in all your application's performance-critical scenarios. As with any aspect of native image use, careful performance measurements are the only way to determine whether hard binding improves your application's performance.

The [DependencyAttribute](#) and [DefaultDependencyAttribute](#) attributes allow you to provide hard binding hints to Ngen.exe.

ⓘ Note

These attributes are hints to Ngen.exe, not commands. Using them does not guarantee hard binding. The meaning of these attributes may change in future releases.

Specifying a binding hint for a dependency

Apply the [DependencyAttribute](#) to an assembly to indicate the likelihood that a specified dependency will be loaded. [LoadHint.Always](#) indicates that hard binding is appropriate, [Default](#) indicates that the default for the dependency should be used, and [Sometimes](#) indicates that hard binding is not appropriate.

The following code shows the attributes for an assembly that has two dependencies. The first dependency (Assembly1) is an appropriate candidate for hard binding, and the second (Assembly2) is not.

C#

```
using System.Runtime.CompilerServices;
[assembly:DependencyAttribute("Assembly1", LoadHint.Always)]
[assembly:DependencyAttribute("Assembly2", LoadHint.Sometimes)]
```

The assembly name does not include the file name extension. Display names can be used.

Specifying a default binding hint for an assembly

Default binding hints are only needed for assemblies that will be used immediately and frequently by any application that has a dependency on them. Apply the [DefaultDependencyAttribute](#) with [LoadHint.Always](#) to such assemblies to specify that hard binding should be used.

ⓘ Note

There is no reason to apply [DefaultDependencyAttribute](#) to .dll assemblies that do not fall into this category, because applying the attribute with any value other than [LoadHint.Always](#) has the same effect as not applying the attribute at all.

Microsoft uses the [DefaultDependencyAttribute](#) to specify that hard binding is the default for a very small number of assemblies in the .NET Framework, such as mscorelib.dll.

Deferred processing

Generation of native images for a very large application can take considerable time. Similarly, changes to a shared component or changes to computer settings might require many native images to be updated. The `install` and `update` actions have a `/queue` option that queues the operation for deferred execution by the native image

service. In addition, Ngen.exe has `queue` and `executeQueuedItems` actions that provide some control over the service. For more information, see [Native Image Service](#).

Native images and JIT compilation

If Ngen.exe encounters any methods in an assembly that it cannot generate, it excludes them from the native image. When the runtime executes this assembly, it reverts to JIT compilation for the methods that were not included in the native image.

In addition, native images are not used if the assembly has been upgraded, or if the image has been invalidated for any reason.

Invalid images

When you use Ngen.exe to create a native image of an assembly, the output depends upon the command-line options that you specify and certain settings on your computer. These settings include the following:

- The version of .NET Framework.
- The exact identity of the assembly (recompilation changes identity).
- The exact identity of all assemblies that the assembly references (recompilation changes identity).
- Security factors.

Ngen.exe records this information when it generates a native image. When you execute an assembly, the runtime looks for the native image generated with options and settings that match the computer's current environment. The runtime reverts to JIT compilation of an assembly if it cannot find a matching native image. The following changes to a computer's settings and environment cause native images to become invalid:

- The version of .NET Framework.

If you apply an update to .NET Framework, all native images that you have created using Ngen.exe become invalid. For this reason, all updates of .NET Framework execute the `Ngen Update` command, to ensure that all native images are regenerated. .NET Framework automatically creates new native images for the .NET Framework libraries that it installs.

- The exact identity of the assembly.

If you recompile an assembly, the assembly's corresponding native image becomes invalid.

- The exact identity of any assemblies the assembly references.

If you update a managed assembly, all native images that directly or indirectly depend on that assembly become invalid and need to be regenerated. This includes both ordinary references and hard-bound dependencies. Whenever a software update is applied, the installation program should execute an `Ngen Update` command to ensure that all dependent native images are regenerated.

- Security factors.

Changing machine security policy to restrict permissions previously granted to an assembly can cause a previously compiled native image for that assembly to become invalid.

For detailed information about how the common language runtime administers code access security and how to use permissions, see [Code Access Security](#).

Troubleshooting

The following troubleshooting topics allow you to see which native images are being used and which cannot be used by your application, to determine when the JIT compiler starts to compile a method, and shows how to opt out of native image compilation of specified methods.

Assembly Binding Log Viewer

To confirm that native images are being used by your application, you can use the [Fuslogvw.exe \(Assembly Binding Log Viewer\)](#). Select **Native Images** in the **Log Categories** box on the binding log viewer window. Fuslogvw.exe provides information about why a native image was rejected.

The `JITCompilationStart` managed debugging assistant

You can use the [`jitCompilationStart`](#) managed debugging assistant (MDA) to determine when the JIT compiler starts to compile a function.

Opting out of native image generation

In some cases, NGen.exe may have difficulty generating a native image for a specific method, or you may prefer that the method be JIT compiled rather than compiled to a native image. In this case, you can use the `System.Runtime.BypassNGenAttribute` attribute to prevent NGen.exe from generating a native image for a particular method. The attribute must be applied individually to each method whose code you do not want to include in the native image. NGen.exe recognizes the attribute and does not generate code in the native image for the corresponding method.

Note, however, that `BypassNGenAttribute` is not defined as a type in the .NET Framework Class Library. In order to consume the attribute in your code, you must first define it as follows:

```
C#  
  
namespace System.Runtime  
{  
    [AttributeUsage(AttributeTargets.Method |  
                    AttributeTargets.Constructor |  
                    AttributeTargets.Property)]  
    public class BypassNGenAttribute : Attribute  
    {  
    }  
}
```

You can then apply the attribute on a per-method basis. The following example instructs the Native Image Generator that it should not generate a native image for the `ExampleClass.ToJITCompile` method.

```
C#  
  
using System;  
using System.Runtime;  
  
public class ExampleClass  
{  
    [BypassNGen]  
    public void ToJITCompile()  
    {  
    }  
}
```

Examples

The following command generates a native image for `ClientApp.exe`, located in the current directory, and installs the image in the native image cache. If a configuration file

exists for the assembly, Ngen.exe uses it. In addition, native images are generated for any .dll files that `ClientApp.exe` references.

Console

```
ngen install ClientApp.exe
```

An image installed with Ngen.exe is also called a root. A root can be an application or a shared component.

The following command generates a native image for `MyAssembly.exe` with the specified path.

Console

```
ngen install c:\myfiles\MyAssembly.exe
```

When locating assemblies and their dependencies, Ngen.exe uses the same probing logic used by the common language runtime. By default, the directory that contains `ClientApp.exe` is used as the application base directory, and all assembly probing begins in this directory. You can override this behavior by using the `/AppBase` option.

ⓘ Note

This is a change from Ngen.exe behavior in the .NET Framework versions 1.0 and 1.1, where the application base is set to the current directory.

An assembly can have a dependency without a reference, for example if it loads a .dll file by using the `Assembly.Load` method. You can create a native image for such a .dll file by using configuration information for the application assembly, with the `/ExeConfig` option. The following command generates a native image for `MyLib.dll`, using the configuration information from `MyApp.exe`.

Console

```
ngen install c:\myfiles\MyLib.dll /ExeConfig:c:\myapps\MyApp.exe
```

Assemblies installed in this way are not removed when the application is removed.

To uninstall a dependency, use the same command-line options that were used to install it. The following command uninstalls the `MyLib.dll` from the previous example.

Console

```
ngen uninstall c:\myfiles\MyLib.dll /ExeConfig:c:\myapps\MyApp.exe
```

To create a native image for an assembly in the global assembly cache, use the display name of the assembly. For example:

Console

```
ngen install "ClientApp, Version=1.0.0.0, Culture=neutral,  
PublicKeyToken=3c7ba247adcd2081, processorArchitecture=MSIL"
```

NGen.exe generates a separate set of images for each scenario you install. For example, the following commands install a complete set of native images for normal operation, another complete set for debugging, and a third for profiling:

Console

```
ngen install MyApp.exe  
ngen install MyApp.exe /debug  
ngen install MyApp.exe /profile
```

Displaying the Native Image Cache

Once native images are installed in the cache, they can be displayed using Ngen.exe. The following command displays all native images in the native image cache.

Console

```
ngen display
```

The `display` action lists all the root assemblies first, followed by a list of all the native images on the computer.

Use the simple name of an assembly to display information only for that assembly. The following command displays all native images in the native image cache that match the partial name `MyAssembly`, their dependencies, and all roots that have a dependency on `MyAssembly`:

Console

```
ngen display MyAssembly
```

Knowing what roots depend on a shared component assembly is useful in gauging the impact of an `update` action after the shared component is upgraded.

If you specify an assembly's file extension, you must either specify the path or execute Ngen.exe from the directory containing the assembly:

```
Console
```

```
ngen display c:\myApps\MyAssembly.exe
```

The following command displays all native images in the native image cache with the name `MyAssembly` and the version 1.0.0.0.

```
Console
```

```
ngen display "myAssembly, version=1.0.0.0"
```

Updating Images

Images are typically updated after a shared component has been upgraded. To update all native images that have changed, or whose dependencies have changed, use the `update` action with no arguments.

```
Console
```

```
ngen update
```

Updating all images can be a lengthy process. You can queue the updates for execution by the native image service by using the `/queue` option. For more information on the `/queue` option and installation priorities, see [Native Image Service](#).

```
Console
```

```
ngen update /queue
```

Uninstalling Images

Ngen.exe maintains a list of dependencies, so that shared components are removed only when all assemblies that depend on them have been removed. In addition, a shared component is not removed if it has been installed as a root.

The following command uninstalls all scenarios for the root `ClientApp.exe`:

```
Console  
ngen uninstall ClientApp
```

The `uninstall` action can be used to remove specific scenarios. The following command uninstalls all debug scenarios for `clientApp.exe`:

```
Console  
ngen uninstall ClientApp /debug
```

ⓘ Note

Uninstalling `/debug` scenarios does not uninstall a scenario that includes both `/profile` and `/debug`.

The following command uninstalls all scenarios for a specific version of `ClientApp.exe`:

```
Console  
ngen uninstall "ClientApp, Version=1.0.0.0"
```

The following commands uninstall all scenarios for `"ClientApp, Version=1.0.0.0, Culture=neutral, PublicKeyToken=3c7ba247adcd2081, processorArchitecture=MSIL"`, or just the debug scenario for that assembly:

```
Console  
ngen uninstall "ClientApp, Version=1.0.0.0, Culture=neutral,  
PublicKeyToken=3c7ba247adcd2081, processorArchitecture=MSIL"  
ngen uninstall "ClientApp, Version=1.0.0.0, Culture=neutral,  
PublicKeyToken=3c7ba247adcd2081, processorArchitecture=MSIL" /debug
```

As with the `install` action, supplying an extension requires either executing Ngen.exe from the directory containing the assembly or specifying a full path.

For examples relating to the native image service, see [Native Image Service](#).

Native Image Task

The native image task is a Windows task that generates and maintains native images. The native image task generates and reclaims native images automatically for supported scenarios. It also enables installers to use [Ngen.exe \(Native Image Generator\)](#) to create and update native images at a deferred time.

The native image task is registered once for each CPU architecture supported on a computer, to allow compilation for applications that target each architecture:

[+] Expand table

Task name	32-bit computer	64-bit computer
NET Framework NGEN v4.0.30319	Yes	Yes
NET Framework NGEN v4.0.30319 64	No	Yes

The native image task is available in .NET Framework 4.5 and later versions, when running on Windows 8 or later. On earlier versions of Windows, the .NET Framework uses the [Native Image Service](#).

Task Lifetime

In general, the Windows Task Scheduler starts the native image task every night when the computer is idle. The task checks for any deferred work that is queued by application installers, any deferred native image update requests, and any automatic image creation. The task completes outstanding work items and then shuts down. If the computer stops being idle while the task is running, the task stops.

You can also start the native image task manually through the Task Scheduler UI or through manual calls to NGen.exe. If the task is started through either of these methods, it will continue running when the computer is no longer idle. Images created manually by using NGen.exe are prioritized to enable predictable behavior for application installers.

Native Image Service

The native image service is a Windows service that generates and maintains native images. The native image service allows the developer to defer the installation and update of native images to periods when the computer is idle.

Normally, the native image service is initiated by the installation program (installer) for an application or update. For priority 3 actions, the service executes during idle time on

the computer. The service saves its state and is capable of continuing through multiple reboots if necessary. Multiple image compilations can be queued.

The service also interacts with the manual Ngen.exe command. Manual commands take precedence over background activity.

Note

On Windows Vista, the name displayed for the native image service is "Microsoft.NET Framework NGEN v2.0.50727_X86" or "Microsoft.NET Framework NGEN v2.0.50727_X64". On all earlier versions of Microsoft Windows, the name is ".NET Runtime Optimization Service v2.0.50727_X86" or ".NET Runtime Optimization Service v2.0.50727_X64".

Launching Deferred Operations

Before beginning an installation or upgrade, pausing the service is recommended. This ensures that the service does not execute while the installer is copying files or putting assemblies in the global assembly cache. The following Ngen.exe command line pauses the service:

```
Console
```

```
ngen queue pause
```

When all deferred operations have been queued, the following command allows the service to resume:

```
Console
```

```
ngen queue continue
```

To defer native image generation when installing a new application or when updating a shared component, use the `/queue` option with the `install` or `update` actions. The following Ngen.exe command lines install a native image for a shared component and perform an update of all roots that may have been affected:

```
Console
```

```
ngen install MyComponent /queue  
ngen update /queue
```

The `update` action regenerates all native images that have been invalidated, not just those that use `MyComponent`.

If your application consists of many roots, you can control the priority of the deferred actions. The following commands queue the installation of three roots. `Assembly1` is installed first, without waiting for idle time. `Assembly2` is also installed without waiting for idle time, but after all priority 1 actions have completed. `Assembly3` is installed when the service detects that the computer is idle.

Console

```
ngen install Assembly1 /queue:1
ngen install Assembly2 /queue:2
ngen install Assembly3 /queue:3
```

You can force queued actions to occur synchronously by using the `executeQueuedItems` action. If you supply the optional priority, this action affects only the queued actions that have equal or lower priority. The default priority is 3, so the following Ngen.exe command processes all queued actions immediately, and does not return until they are finished:

Console

```
ngen executeQueuedItems
```

Synchronous commands are executed by Ngen.exe and do not use the native image service. You can execute actions using Ngen.exe while the native image service is running.

Service Shutdown

After being initiated by the execution of an Ngen.exe command that includes the `/queue` option, the service runs in the background until all actions have been completed. The service saves its state so that it can continue through multiple reboots if necessary. When the service detects that there are no more actions queued, it resets its status so that it will not restart the next time the computer is booted, and then it shuts itself down.

Service Interaction with Clients

In .NET Framework version 2.0, the only interaction with the native image service is through the command-line tool Ngen.exe. Use the command-line tool in installation scripts to queue actions for the native image service and to interact with the service.

See also

- [Tools](#)
- [Managed Execution Process](#)
- [How the Runtime Locates Assemblies](#)
- [Developer command-line shells](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Peverify.exe (PEVerify tool)

Article • 07/23/2022

The PEVerify tool helps developers who generate Microsoft intermediate language (MSIL) (such as compiler writers and script engine developers) to determine whether their MSIL code and associated metadata meet type safety requirements. Some compilers generate verifiably type-safe code only if you avoid using certain language constructs. If you're using such a compiler, you may want to verify that you have not compromised the type safety of your code. You can run the PEVerify tool on your files to check the MSIL and metadata.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

Syntax

Console

```
peverify filename [options]
```

Parameters

[] Expand table

Argument	Description
<i>filename</i>	The portable executable (PE) file for which to check the MSIL and metadata.

[] Expand table

Option	Description
/break= <i>maxErrorCount</i>	Aborts verification after <i>maxErrorCount</i> errors. This parameter is not supported in .NET Framework version 2.0 or later.
/clock	Measures and reports the following verification times in milliseconds: MD Val. cycle Metadata validation cycle MD Val. pure

Option	Description
	Metadata validation pure
	<p>IL Ver. cycle Microsoft intermediate language (MSIL) verification cycle</p>
	<p>IL Ver pure MSIL verification pure</p>
	<p>The MD Val. cycle and IL Ver. cycle times include the time required to perform necessary startup and shutdown procedures. The MD Val. pure and IL Ver pure times reflect the time required to perform the validation or verification only.</p>
/help	Displays command syntax and options for the tool.
/HRESULT	Displays error codes in hexadecimal format.
/ignore= <i>hex.code</i> [, <i>hex.code</i>]	Ignores the specified error codes.
/ignore=@ <i>responseFile</i>	Ignores the error codes listed in the specified response file.
/IL	Performs MSIL type safety verification checks for methods implemented in the assembly specified by <i>filename</i> . The tool returns detailed descriptions for each problem found unless you specify the /quiet option.
/MD	Performs metadata validation checks on the assembly specified by <i>filename</i> . This option walks the full metadata structure within the file and reports all validation problems encountered.
/nologo	Suppresses the display of product version and copyright information.
/nosymbols	In .NET Framework version 2.0, suppresses line numbers for backward compatibility.
/quiet	Specifies quiet mode; suppresses output of the verification problem reports. Perverify.exe still reports whether the file is type safe, but does not report information on problems preventing type safety verification.
/transparent	Verify only the transparent methods.
/unique	Ignores repeating error codes.
/verbose	In .NET Framework version 2.0, displays additional information in MSIL verification messages.
/?	Displays command syntax and options for the tool.

Remarks

The common language runtime relies on the type-safe execution of application code to help enforce security and isolation mechanisms. Normally, code that is not **verifiably type safe** cannot run, although you can set security policy to allow the execution of trusted but unverifiable code.

If neither the **/md** nor **/il** options are specified, Peverify.exe performs both types of checks. Peverify.exe performs **/md** checks first. If there are no errors, **/il** checks are made. If you specify both **/md** and **/il**, **/il** checks are made even if there are errors in the metadata. Thus, if there are no metadata errors, **peverify filename** is equivalent to **peverify filename /md /il**.

Peverify.exe performs comprehensive MSIL verification checks based on dataflow analysis plus a list of several hundred rules on valid metadata. For detailed information on the checks Peverify.exe performs, see the "Metadata Validation Specification" and the "MSIL Instruction Set Specification" in the Tools Developers Guide folder in the Windows SDK.

.NET Framework version 2.0 or later supports verifiable `byref` returns specified using the following MSIL instructions: `dup`, `ldslda`, `ldflda`, `ldelema`, `call`, and `unbox`.

Examples

The following command performs metadata validation checks and MSIL type safety verification checks for methods implemented in the assembly `myAssembly.exe`.

Console

```
peverify myAssembly.exe /md /il
```

Upon successful completion of the above request, Peverify.exe displays the following message.

Output

```
All classes and methods in myAssembly.exe Verified
```

The following command performs metadata validation checks and MSIL type safety verification checks for methods implemented in the assembly `myAssembly.exe`. The tool displays the time required to perform these checks.

Console

```
peverify myAssembly.exe /md /il /clock
```

Upon successful completion of the above request, Peverify.exe displays the following message.

Output

```
All classes and methods in myAssembly.exe Verified
Timing: Total run      320 msec
        MD Val.cycle  40 msec
        MD Val.pure   10 msec
        IL Ver.cycle  270 msec
        IL Ver.pure   230 msec
```

The following command performs metadata validation checks and MSIL type safety verification checks for methods implemented in the assembly `myAssembly.exe`.

Peverify.exe stops, however, when it reaches the maximum error count of 100. The tool also ignores the specified error codes.

Console

```
peverify myAssembly.exe /break=100 /ignore=0x12345678,0xABCD1234
```

The following command produces the same result as the above previous example, but specifies the error codes to ignore in the response file `ignoreErrors.rsp`.

Console

```
peverify myAssembly.exe /break=100 /ignore@ignoreErrors.rsp
```

The response file can contain a comma-separated list of error codes.

text

```
0x12345678, 0xABCD1234
```

Alternatively, the response file can be formatted with one error code per line.

text

```
0x12345678
0xABCD1234
```

See also

- [Tools](#)
- [Writing Verifiably Type-Safe Code](#)
- [Type Safety and Security](#)
- [Developer command-line shells](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Regasm.exe (Assembly Registration Tool)

Article • 03/22/2022

The Assembly Registration tool reads the metadata within an assembly and adds the necessary entries to the registry, which allows COM clients to create .NET Framework classes transparently. Once a class is registered, any COM client can use it as though the class were a COM class. The class is registered only once, when the assembly is installed. Instances of classes within the assembly cannot be created from COM until they are actually registered.

To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
regasm assemblyFile [options]
```

Parameters

Parameter	Description
<i>assemblyFile</i>	The assembly to be registered with COM.

Option	Description
/codebase	Creates a Codebase entry in the registry. The Codebase entry specifies the file path for an assembly that's not installed in the global assembly cache. Don't specify this option if you will subsequently install the assembly that you're registering into the global assembly cache. It is strongly recommended the <i>assemblyFile</i> argument that you specify with the /codebase option be a strong-named assembly .
/registered	Specifies that this tool will only refer to type libraries that have already been registered.

Option	Description
/asmpath:directory	Specifies a directory containing assembly references. Must be used with the /regfile option.
/nologo	Suppresses the Microsoft startup banner display.
/regfile [: <i>regFile</i>]	Generates the specified .reg file for the assembly, which contains the needed registry entries. Specifying this option does not change the registry. You cannot use this option with the /u or /tlb options.
/silent or /s	Suppresses the display of success messages.
/tlb [: <i>typeLibFile</i>]	Generates a type library from the specified assembly containing definitions of the accessible types defined within the assembly.
/unregister or /u	Unregisters the creatable classes found in <i>assemblyFile</i> . Omitting this option causes Regasm.exe to register the creatable classes in the assembly.
/verbose	Specifies verbose mode; displays a list of any referenced assemblies for which a type library needs to be generated, when specified with the /tlb option.
/? or /help	Displays command syntax and options for the tool.

ⓘ Note

The Regasm.exe command-line options are case insensitive. You only need to provide enough of the option to uniquely identify it. For example, **/n** is equivalent to **/nologo** and **/t: outfile.tlb** is equivalent to **/tlb: outfile.tlb**.

Remarks

You can use the **/regfile** option to generate a .reg file that contains the registry entries instead of making the changes directly to the registry. You can update the registry on a computer by importing the .reg file with the Registry Editor tool (Regedit.exe). The .reg file does not contain any registry updates that can be made by user-defined register functions. The **/regfile** option only emits registry entries for managed classes. This option does not emit entries for **TypeLibIDs** or **InterfaceIDs**.

When you specify the **/tlb** option, Regasm.exe generates and registers a type library describing the types found in the assembly. Regasm.exe places the generated type libraries in the current working directory or the directory specified for the output file. Generating a type library for an assembly that references other assemblies may cause several type libraries to be generated at once. You can use the type library to provide

type information to development tools like Visual Studio. Don't use the **/tlb** option if the assembly you are registering was produced by the Type Library Importer ([Tlbimp.exe](#)). You cannot export a type library from an assembly that was imported from a type library. Using the **/tlb** option has the same effect as using the Type Library Exporter ([Tlbexp.exe](#)) and Regasm.exe, with the exception that Tlbexp.exe does not register the type library it produces. If you use the **/tlb** option to register a type library, you can use the **/tlb** option with the **/unregister** option to unregister the type library. Using the two options together will unregister the type library and interface entries, which can clean the registry considerably.

When you register an assembly for use by COM, Regasm.exe adds entries to the registry on the local computer. More specifically, it creates version-dependent registry keys that allow multiple versions of the same assembly to run side by side on a computer. The first time an assembly is registered, one top-level key is created for the assembly, and a unique subkey is created for the specific version. Each time you register a new version of the assembly, Regasm.exe creates a subkey for the new version.

For example, consider a scenario where you register the managed component, myComp.dll, version 1.0.0.0 for use by COM. Later, you register myComp.dll, version 2.0.0.0. You determine that all COM client applications on the computer are using myComp.dll version 2.0.0.0 and you decide to unregister myComponent.dll version 1.0.0.0. This registry scheme allows you to unregister myComp.dll version 1.0.0.0 because only the version 1.0.0.0 subkey is removed.

After registering an assembly using Regasm.exe, you can install it in the [global assembly cache](#) so that it can be activated from any COM client. If the assembly is only going to be activated by a single application, you can place it in that application's directory. Using the **/codebase** option is an alternative to using the global assembly cache; however, the location of the assembly during registration is recorded globally and activation will fail if the assembly is moved. If the assembly isn't found through [probing](#), the **/codebase** option will load the assembly in a load-from context that has additional considerations documented in [Assembly.LoadFrom](#).

Examples

The following command registers all public classes contained in `myTest.dll`.

Console

```
regasm myTest.dll
```

The following command generates the file `myTest.reg`, which contains all the necessary registry entries. This command does not update the registry.

Console

```
regasm myTest.dll /regfile:myTest.reg
```

The following command registers all public classes contained in `myTest.dll`, and generates and registers the type library `myTest.tlb`, which contains definitions of all the public types defined in `myTest.dll`.

Console

```
regasm myTest.dll /tlb:myTest.tlb
```

See also

- [Tools](#)
- [Tlbexp.exe \(Type Library Exporter\)](#)
- [Tlbimp.exe \(Type Library Importer\)](#)
- [Registering Assemblies with COM](#)
- [Developer command-line shells](#)

Regsvcs.exe (.NET Services Installation Tool)

Article • 09/15/2021

The .NET Services Installation tool performs the following actions:

- Loads and registers an assembly.
- Generates, registers, and installs a type library into a specified COM+ application.
- Configures services that you have added programmatically to your class.

To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
regsvcs [/c | /fc | /u] [/tlb:typeLibraryFile] [/extlb]
[/reconfig] [/componly] [/appname:applicationName]
[/nologo] [/quiet]assemblyFile.dll
```

Parameters

Argument	Description
<i>assemblyFile.dll</i>	The source assembly file. The assembly must be signed with a strong name. For more information, see Signing an Assembly with a Strong Name .

Option	Description
<i>/appdir: path</i>	Specifies the root directory of the application.
<i>/appname: applicationName</i>	Specifies the name of the COM+ application to either find or create.
<i>/c</i>	Creates the target application.
<i>/componly</i>	Configures components only; ignores methods and interfaces.

Option	Description
/exapp	Specifies to the tool to expect an existing application.
/extlb	Uses an existing type library.
/fc	Finds or creates the target application.
/help	Displays command syntax and options for the tool.
/noreconfig	Does not reconfigure an existing target application.
/nologo	Suppresses the Microsoft startup banner display.
/parname: <i>name</i>	Specifies the name or id of the COM+ application to either find or create.
/reconfig	Reconfigures an existing target application. This is the default.
/tlb: <i>typelibraryfile</i>	Specifies the type library file to install.
/u	Uninstalls the target application.
/quiet	Specifies quiet mode; suppresses the logo and success message display.
/?	Displays command syntax and options for the tool.

Remarks

Regsvcs.exe requires a source assembly file specified by *assemblyFile.dll*. This assembly must be signed with a strong name. For more information on strong name signing, see [Signing an Assembly with a Strong Name](#). The names of the target application and the type library file are optional. The *applicationName* argument can be generated from the source assembly file and will be created by Regsvcs.exe, if it does not already exist. The *typelibraryfile* argument can specify a type library name. If you do not specify a type library name, Regsvcs.exe uses the assembly name as the default.

When Regsvcs.exe registers a component's methods, it is subject to the [demands](#) and [link demands](#) on those methods. Because the tool executes in a fully-trusted environment, most demands for a permission succeed. However, Regsvcs.exe cannot register components with methods protected by a demand or link demand for the [StrongNameIdentityPermission](#) or the [PublisherIdentityPermission](#).

You must have administrative privileges on the local computer to use Regsvcs.exe.

If Regsvcs.exe fails while performing any of these actions, it displays corresponding error messages.

Examples

The following command adds all public classes contained in `myTest.dll` to `myTargetApp` (an existing COM+ application) and produces the `myTest.tlb` type library.

Console

```
regsvcs /appname:myTargetApp myTest.dll
```

The following command adds all public classes contained in `myTest.dll` to `myTargetApp` (an existing COM+ application) and produces the `newTest.tlb` type library.

Console

```
regsvcs /appname:myTargetApp /tlb:newTest.tlb myTest.dll
```

See also

- [Tools](#)
- [How to: Sign an Assembly with a Strong Name](#)
- [Developer command-line shells](#)

Resgen.exe (Resource File Generator)

Article • 10/23/2023

The Resource File Generator (Resgen.exe) converts text (.txt or .restext) files and XML-based resource format (.resx) files to common language runtime binary (.resources) files that can be embedded in a runtime binary executable or satellite assembly. For more information, see [Create resource files](#).

Resgen.exe is a general-purpose resource conversion utility that performs the following tasks:

- Converts .txt or .restext files to .resources or .resx files. (The format of .restext files is identical to the format of .txt files. However, the .restext extension helps you identify text files that contain resource definitions more easily.)
- Converts .resources files to text or .resx files.
- Converts .resx files to text or .resources files.
- Extracts the string resources from an assembly into a .resw file that is suitable for use in a Windows 8.x Store app.
- Creates a strongly typed class that provides access to individual named resources and to the [ResourceManager](#) instance.

If Resgen.exe fails for any reason, the return value is –1.

To get help with Resgen.exe, you can use the following command, with no options specified, to display the command syntax and options for Resgen.exe:

```
Console
resgen
```

You can also use the `/?` switch:

```
Console
resgen /?
```

If you use Resgen.exe to generate binary .resources files, you can use a language compiler to embed the binary files into executable assemblies, or you can use the [Assembly Linker \(Al.exe\)](#) to compile them into satellite assemblies.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
resgen [-define:symbol1[,symbol2,...]] [/useSourcePath] filename.extension  
| /compile filename.extension... [outputFilename.extension] [/r:assembly]  
[/str:lang[,namespace[,class[,file]]]] [/publicclass]
```

Console

```
resgen filename.extension [outputDirectory]
```

Parameters

Parameter or switch	Description
<code>/define: symbol1[, symbol2,...]</code>	<p>Starting with .NET Framework 4.5, supports conditional compilation in text-based (.txt or .restext) resource files. If <i>symbol</i> corresponds to a symbol included in the input text file within a <code>#ifdef</code> construct, the associated string resource is included in the .resources file. If the input text file includes an <code>#if !</code> statement with a symbol that is not defined by the <code>/define</code> switch, the associated string resource is included in the resources file.</p> <p><code>/define</code> is ignored if it is used with non-text files. Symbols are case-sensitive.</p> <p>For more information about this option, see Conditionally Compiling Resources later in this topic.</p>
<code>useSourcePath</code>	Specifies that the input file's current directory is to be used to resolve relative file paths.
<code>/compile</code>	Enables you to specify multiple .resx or text files to convert to multiple .resources files in a single bulk operation. If you do not specify this option,

Parameter or switch	Description
	<p>you can specify only one input file argument. Output files are named <i>filename.resources</i>.</p>
	<p>This option cannot be used with the <code>/str:</code> option.</p>
	<p>For more information about this option, see Compiling or Converting Multiple Files later in this topic.</p>
<code>/r: assembly</code>	<p>References metadata from the specified assembly. It is used when converting .resx files and allows Resgen.exe to serialize or deserialize object resources. It is similar to the <code>/reference:</code> or <code>/r:</code> options for the C# and Visual Basic compilers.</p>
<code>filename.extension</code>	<p>Specifies the name of the input file to convert. If you're using the first, lengthier command-line syntax presented before this table, <code>extension</code> must be one of the following:</p>
	<p>.txt or .restext A text file to convert to a .resources or a .resx file. Text files can contain only string resources. For information about the file format, see the "Resources in Text Files" section of Create resource files.</p>
	<p>.resx An XML-based resource file to convert to a .resources or a text (.txt or .restext) file.</p>
	<p>.resources A binary resource file to convert to a .resx or a text (.txt or .restext) file.</p>
	<p>If you're using the second, shorter command-line syntax presented before this table, <code>extension</code> must be the following:</p>
	<p>.exe or .dll A .NET Framework assembly (executable or library) whose string resources are to be extracted to a .resw file for use in developing Windows 8.x Store apps.</p>

Parameter or switch	Description
<code>outputFilename.extension</code>	<p>Specifies the name and type of the resource file to create.</p> <p>This argument is optional when converting from a .txt, .restext, or .resx file to a .resources file. If you do not specify <code>outputFilename</code>, Resgen.exe appends a .resources extension to the input <code>filename</code> and writes the file to the directory that contains <code>filename,extension</code>.</p> <p>The <code>outputFilename.extension</code> argument is mandatory when converting from a .resources file. Specify a file name with the .resx extension when converting a .resources file to an XML-based resource file. Specify a file name with the .txt or .restext extension when converting a .resources file to a text file. You should convert a .resources file to a .txt file only when the .resources file contains only string values.</p>
<code>outputDirectory</code>	<p>For Windows 8.x Store apps, specifies the directory in which a .resw file that contains the string resources in <code>filename.extension</code> will be written. <code>outputDirectory</code> must already exist.</p>
<code>/str:</code> <code>language[,namespace[,classname[,filename]]]</code>	<p>Creates a strongly typed resource class file in the programming language specified in the <code>language</code> option. <code>language</code> can consist of one of the following literals:</p> <ul style="list-style-type: none"> - For C#: <code>c#</code>, <code>cs</code>, or <code>csharp</code>. - For Visual Basic: <code>vb</code> or <code>visualbasic</code>. - For VBScript: <code>vbs</code> or <code>vbscript</code>. - For C++: <code>c++</code>, <code>mc</code>, or <code>cpp</code>. - For JavaScript: <code>js</code>, <code>jscript</code>, or <code>javascript</code>. <p>The <code>namespace</code> option specifies the project's default namespace, the <code>classname</code> option specifies the name of the generated class, and the <code>filename</code> option specifies the name of the class file.</p>
	<p>The <code>/str:</code> option allows only one input file, so it cannot be used with the <code>/compile</code> option.</p> <p>If <code>namespace</code> is specified but <code>classname</code> is not, the class name is derived from the output file</p>

Parameter or switch	Description
	<p>name (for example, underscores are substituted for periods). The strongly typed resources might not work correctly as a result. To avoid this, specify both class name and output file name.</p> <p>For more information about this option, see Generating a Strongly Typed Resource Class later in this topic.</p>
<code>/publicClass</code>	<p>Creates a strongly typed resource class as a public class. By default, the resource class is <code>internal</code> in C# and <code>Friend</code> in Visual Basic.</p> <p>This option is ignored if the <code>/str:</code> option is not used.</p>

Resgen.exe and Resource File Types

In order for Resgen.exe to successfully convert resources, text and .resx files must follow the correct format.

Text (.txt and .restext) Files

Text (.txt or .restext) files may contain only string resources. String resources are useful if you are writing an application that must have strings translated into several languages. For example, you can easily regionalize menu strings by using the appropriate string resource. Resgen.exe reads text files that contain name/value pairs, where the name is a string that describes the resource and the value is the resource string itself.

ⓘ Note

For information about the format of .txt and .restext files, see the "Resources in Text Files" section of [Create resource files](#).

A text file that contains resources must be saved with UTF-8 or Unicode (UTF-16) encoding unless it contains only characters in the Basic Latin range (to U+007F). Resgen.exe removes extended ANSI characters when it processes a text file that is saved using ANSI encoding.

Resgen.exe checks the text file for duplicate resource names. If the text file contains duplicate resource names, Resgen.exe will emit a warning and ignore the second value.

.resx Files

The .resx resource file format consists of XML entries. You can specify string resources within these XML entries, as you would in text files. A primary advantage of .resx files over text files is that you can also specify or embed objects. When you view a .resx file, you can see the binary form of an embedded object (for example, a picture) when this binary information is a part of the resource manifest. As with text files, you can open a .resx file with a text editor (such as Notepad or Microsoft Word) and write, parse, and manipulate its contents. Note that this requires a good knowledge of XML tags and the .resx file structure. For more details on the .resx file format, see the "Resources in .resx Files" section of [Create resource files](#).

In order to create a .resources file that contains embedded nonstring objects, you must either use Resgen.exe to convert a .resx file containing objects or add the object resources to your file directly from code by calling the methods provided by the [ResourceWriter](#) class.

If your .resx or .resources file contains objects and you use Resgen.exe to convert it to a text file, all the string resources will be converted correctly, but the data types of the nonstring objects will also be written to the file as strings. You will lose the embedded objects in the conversion, and Resgen.exe will report that an error occurred in retrieving the resources.

Converting Between Resources File Types

When you convert between different resource file types, Resgen.exe may not be able to perform the conversion or may lose information about specific resources, depending on the source and target file types. The following table specifies the types of conversions that are successful when converting from one resource file type to another.

Convert from	To text file	To .resx file	To .resw file	To .resources file
Text (.txt or .restext) file	--	No issues	Not supported	No issues
.resx file	Conversion fails if file contains non-string resources (including file links)	--	Not supported*	No issues
.resources file	Conversion fails if file contains non-string	No issues	Not supported	--

Convert from	To text file	To .resx file	To .resw file	To .resources file
resources (including file links)				
.exe or .dll assembly	Not supported	Not supported	Only string resources (including path names) are recognized as resources	Not supported

*In Windows 8.x Store apps, you use .resw files to create resources. Despite the difference of file extension, the .resw file format is identical to the .resx file format, except that .resw files can contain only strings and file paths. You can convert .resx files that contain only strings and file paths into .resw files by simply changing the file extension.

Performing Specific Resgen.exe Tasks

You can use Resgen.exe in diverse ways: to compile a text-based or XML-based resource file into a binary file, to convert between resource file formats, and to generate a class that wraps [ResourceManager](#) functionality and provides access to resources. This section provides detailed information about each task:

- [Compiling Resources into a Binary File](#)
- [Converting Between Resource File Types](#)
- [Compiling or Converting Multiple Files](#)
- [Exporting Resources to a .resw File](#)
- [Conditionally Compiling Resources](#)
- [Generating a Strongly Typed Resource Class](#)

Compiling Resources into a Binary File

The most common use of Resgen.exe is to compile a text-based resource file (a .txt or .restext file) or an XML-based resource file (a .resx file) into a binary .resources file. The output file then can be embedded in a main assembly by a language compiler or in a satellite assembly by [Assembly Linker \(AL.exe\)](#).

The syntax to compile a resource file is:

Console

```
resgen inputFilename [outputFilename]
```

where the parameters are:

`inputFilename` The file name, including the extension, of the resource file to compile.

Resgen.exe only compiles files with extensions of .txt, .restext, or .resx.

`outputFilename` The name of the output file. If you omit `outputFilename`, Resgen.exe creates a .resources file with the root file name of `inputFilename` in the same directory as `inputFilename`. If `outputFilename` includes a directory path, the directory must exist.

You provide a fully qualified namespace for the .resources file by specifying it in the file name and separating it from the root file name by a period. For example, if

`outputFilename` is `MyCompany.Libraries.Strings.resources`, the namespace is

`MyCompany.Libraries`.

The following command reads the name/value pairs in Resources.txt and writes a binary .resources file named Resources.resources. Because the output file name is not specified explicitly, it receives the same name as the input file by default.

Console

```
resgen Resources.txt
```

The following command reads the name/value pairs in Resources.restext and writes a binary resources file named StringResources.resources.

Console

```
resgen Resources.restext StringResources.resources
```

The following command reads an XML-based input file named Resources.resx and writes a binary .resources file named Resources.resources.

Console

```
resgen Resources.resx Resources.resources
```

Converting Between Resource File Types

In addition to compiling text-based or XML-based resource files into binary .resources files, Resgen.exe can convert any supported file type to any other supported file type. This means that it can perform the following conversions:

- .txt and .restext files to .resx files.
- .resx files to .txt and .restext files.
- .resources files to .txt and .restext files.
- .resources files to .resx files.

The syntax is the same as that shown in the previous section.

In addition, you can use Resgen.exe to convert embedded resources in a .NET Framework assembly to a .resw file for Windows 8.x Store apps.

The following command reads a binary resources file Resources.resources and writes an XML-based output file named Resources.resx.

```
Console  
resgen Resources.resources Resources.resx
```

The following command reads a text-based resources file named StringResources.txt and writes an XML-based resources file named LibraryResources.resx. In addition to containing string resources, the .resx file could also be used to store non-string resources.

```
Console  
resgen StringResources.txt LibraryResources.resx
```

The following two commands read an XML-based resources file named Resources.resx and write text files named Resources.txt and Resources.restext. Note that if the .resx file contains any embedded objects, they will not be accurately converted into the text files.

```
Console  
resgen Resources.resx Resources.txt  
resgen Resources.resx Resources.restext
```

Compiling or Converting Multiple Files

You can use the `/compile` switch to convert a list of resource files from one format to another in a single operation. The syntax is:

Console

```
resgen /compile filename.extension [filename.extension...]
```

The following command compiles three files, `StringResources.txt`, `TableResources.resw`, and `ImageResources.resw`, into separate `.resources` files named `StringResources.resources`, `TableResources.resources`, and `ImageResources.resources`.

Console

```
resgen /compile StringResources.txt TableResources.resx ImageResources.resx
```

Exporting Resources to a `.resw` File

If you're developing a Windows 8.x Store app, you may want to use resources from an existing desktop app. However, the two kinds of applications support different file formats. In desktop apps, resources in text (`.txt` or `.restext`) or `.resx` files are compiled into binary `.resources` files. In Windows 8.x Store apps, `.resw` files are compiled into binary package resource index (PRI) files. You can use `Resgen.exe` to bridge this gap by extracting resources from an executable or a satellite assembly and writing them to one or more `.resw` files that can be used when developing a Windows 8.x Store app.

ⓘ Important

Visual Studio automatically handles all conversions necessary for incorporating the resources in a portable library into a Windows 8.x Store app. Using `Resgen.exe` directly to convert the resources in an assembly to `.resw` file format is of interest only to developers who want to develop a Windows 8.x Store app outside of Visual Studio.

The syntax to generate `.resw` files from an assembly is:

Console

```
resgen filename.extension [outputDirectory]
```

where the parameters are:

`filename.extension` The name of a .NET Framework assembly (an executable or .DLL). If the file contains no resources, Resgen.exe does not create any files.

`outputDirectory` The existing directory to which to write the .resw files. If `outputDirectory` is omitted, .resw files are written to the current directory. Resgen.exe creates one .resw file for each .resources file in the assembly. The root file name of the .resw file is the same as the root name of the .resources file.

The following command creates a .resw file in the Win8Resources directory for each .resources file embedded in MyApp.exe:

```
Console  
resgen MyApp.exe Win8Resources
```

Conditionally Compiling Resources

Starting with .NET Framework 4.5, Resgen.exe supports conditional compilation of string resources in text (.txt and .restext) files. This enables you to use a single text-based resource file in multiple build configurations.

In a .txt or .restext file, you use the `#ifdef ... #endif` construct to include a resource in the binary .resources file if a symbol is defined, and you use the `#if ! ... #endif` construct to include a resource if a symbol is not defined. At compile time, you then define symbols by using the `/define:` option followed by a comma-delimited list of symbols. The comparison is case-sensitive. The case of symbols defined by `/define` must match the case of symbols in the text files to be compiled.

For example, the following file named UIResources.text includes a string resource named `AppTitle` that can take one of three values, depending on whether symbols named `PRODUCTION`, `CONSULT`, or `RETAIL` are defined.

```
text  
  
#ifdef PRODUCTION  
AppTitle=My Software Company Project Manager  
#endif  
#ifdef CONSULT  
AppTitle=My Consulting Company Project Manager  
#endif  
#ifdef RETAIL  
AppTitle=My Retail Store Project Manager  
#endif  
FileMenuItemName=File
```

The file can then be compiled into a binary .resources file with the following command:

Console

```
resgen /define:CONSULT UIResources.restext
```

This produces a .resources file that contains two string resources. The value of the `AppTitle` resource is "My Consulting Company Project Manager".

Generating a Strongly Typed Resource Class

Resgen.exe supports strongly typed resources, which encapsulates access to resources by creating classes that contain a set of static read-only properties. This provides an alternative to calling the methods of the [ResourceManager](#) class directly to retrieve resources. You can enable strongly typed resource support by using the `/str` option in Resgen.exe, which wraps the functionality of the [StronglyTypedResourceBuilder](#) class. When you specify the `/str` option, the output of Resgen.exe is a class that contains strongly typed properties that match the resources that are referenced in the input parameter. This class provides strongly typed read-only access to the resources that are available in the file processed.

The syntax to create a strongly typed resource is:

Console

```
resgen inputFilename [outputFilename] /str:language[,namespace,  
[classname[,filename]]] [/publicClass]
```

The parameters and switches are:

`inputFilename` The file name, including the extension, of the resource file for which to generate a strongly typed resource class. The file can be a text-based, XML-based, or binary .resources file; it can have an extension of .txt, .restext, .resw, or .resources.

`outputFilename` The name of the output file. If `outputFilename` includes a directory path, the directory must exist. If you omit `outputFilename`, Resgen.exe creates a .resources file with the root file name of `inputFilename` in the same directory as `inputFilename`.

`outputFilename` can be a text-based, XML-based, or binary .resources file. If the file extension of `outputFilename` is different from the file extension of `inputFilename`, Resgen.exe performs the file conversion.

If `inputFilename` is a .resources file, Resgen.exe copies the .resources file if `outputFilename` is also a .resources file. If `outputFilename` is omitted, Resgen.exe overwrites `inputFilename` with an identical .resources file.

language The language in which to generate source code for the strongly typed resource class. Possible values are `cs`, `C#`, and `csharp` for C# code, `vb` and `visualbasic` for Visual Basic code, `vbs` and `vbscript` for VBScript code, and `c++`, `mc`, and `cpp` for C++ code.

namespace The namespace that contains the strongly typed resource class. The .resources file and the resource class should have the same namespace. For information about specifying the namespace in the `outputFilename`, see [Compiling Resources into a Binary File](#). If *namespace* is omitted, the resource class is not contained in a namespace.

classname The name of the strongly typed resource class. This should correspond to the root name of the .resources file. For example, if Resgen.exe generates a .resources file named MyCompany.Libraries.Strings.resources, the name of the strongly typed resource class is Strings. If *classname* is omitted, the generated class is derived from the root name of `outputFilename`. If `outputFilename` is omitted, the generated class is derived from the root name of `inputFilename`.

classname cannot contain invalid characters such as embedded spaces. If *classname* contains embedded spaces, or if *classname* is generated by default from *inputFilename*, and *inputFilename* contains embedded spaces, Resgen.exe replaces all invalid characters with an underscore (`_`).

filename The name of the class file.

`/publicclass` Makes the strongly typed resource class public rather than `internal` (in C#) or `Friend` (in Visual Basic). This allows the resources to be accessed from outside the assembly in which they are embedded.

Important

When you create a strongly typed resource class, the name of your .resources file must match the namespace and class name of the generated code. However, Resgen.exe allows you to specify options that produce a .resources file that has an incompatible name. To work around this behavior, rename the output file after it has been generated.

The strongly typed resource class has the following members:

- A parameterless constructor, which can be used to instantiate the strongly typed resource class.
- A `static` (C#) or `Shared` (Visual Basic) and read-only `ResourceManager` property, which returns the `ResourceManager` instance that manages the strongly typed resource.
- A static `Culture` property, which allows you to set the culture used for resource retrieval. By default, its value is `null`, which means that the current UI culture is used.
- One `static` (C#) or `Shared` (Visual Basic) and read-only property for each resource in the .resources file. The name of the property is the name of the resource.-

For example, the following command compiles a resource file named StringResources.txt into StringResources.resources and generates a class named `StringResources` in a Visual Basic source code file named StringResources.vb that can be used to access the Resource Manager.

Console

```
resgen StringResources.txt /str:vb,,StringResources
```

See also

- [Tools](#)
- [Resources in .NET apps](#)
- [Create resource files](#)
- [Al.exe \(Assembly Linker\)](#)
- [Developer command-line shells](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

The .NET documentation is open source. Provide feedback here.

 [Open a documentation issue](#)

 [Provide product feedback](#)

SecAnnotate.exe (.NET Security Annotator Tool)

Article • 03/18/2022

The .NET Security Annotator tool (SecAnnotate.exe) is a command-line application that identifies the `SecurityCritical` and `SecuritySafeCritical` portions of one or more assemblies.

A Visual Studio extension, [Security Annotator](#), provides a graphical user interface to SecAnnotate.exe and enables you to run the tool from Visual Studio.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following, where *parameters* are described in the following section, and *assemblies* consist of one or more assembly names separated by blanks:

Syntax

Console

```
SecAnnotate.exe [parameters] [assemblies]
```

Parameters

Option	Description
<code>/a</code>	Shows statistics about the use of transparency in assemblies that are being analyzed.
or	
<code>/showstatistics</code>	
<code>/d: directory</code>	Specifies a directory to search for dependent assemblies during annotation.
or	
<code>/referencedir: directory</code>	

Option	Description
<p>/i</p> <p>or</p> <p>/includesignatures</p>	<p>Includes extended signature information in the annotation report file.</p>
<p>/n</p> <p>or</p> <p>/nogac</p>	<p>Suppresses searching for referenced assemblies in the global assembly cache.</p>
<p>/o: <i>output.xml</i></p> <p>or</p> <p>/out: <i>output.xml</i></p>	<p>Specifies the output annotation file.</p>
<p>/p: <i>maxpasses</i></p> <p>or</p> <p>/maximumpasses: <i>maxpasses</i></p>	<p>Specifies the maximum number of annotation passes to make on assemblies before stopping the generation of new annotations.</p>
<p>/q</p> <p>or</p> <p>/quiet</p>	<p>Specifies quiet mode, in which the annotator does not output status messages; it outputs only error information.</p>
<p>/r: <i>assembly</i></p> <p>or</p> <p>/referenceassembly: <i>assembly</i></p>	<p>Includes the specified assembly when resolving dependent assemblies during annotation. Reference assemblies are given priority over assemblies that are found in the reference path.</p>
<p>/s: <i>rulename</i></p> <p>or</p> <p>/suppressrule: <i>rulename</i></p>	<p>Suppresses running the specified transparency rule on the input assemblies.</p>

Option	Description
<code>/t</code> or <code>/forcetransparent</code>	Forces the Annotator tool to treat all assemblies that do not have any transparency annotations as if they were entirely transparent.
<code>/t:assembly</code> or <code>/forcetransparent:assembly</code>	Force the given assembly to be transparent, regardless of its current assembly-level annotations.
<code>/v</code> or <code>/verify</code>	Verifies only that an assembly's annotations are correct; does not attempt to make multiple passes to find all required annotations if the assembly does not verify.
<code>/x</code> or <code>/verbose</code>	Specifies verbose output while annotating.
<code>/y: directory</code> or <code>/symbolpath: directory</code>	Includes the specified directory when searching for symbol files during annotation.

Remarks

Parameters and assemblies may also be provided in a response file that is specified on the command line and prefixed with an at sign (@). Each line in the response file should contain a single parameter or assembly name.

For more information about the .NET Security Annotator, see the entry [Using SecAnnotate to Analyze Your Assemblies for Transparency Violations](#) in the .NET Security blog.

Examples

SignTool.exe (Sign Tool)

Article • 03/30/2023

Sign Tool is a command-line tool that digitally signs files, verifies signatures in files, and time-stamps files.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

ⓘ Note

The Windows 10 SDK, Windows 10 HLK, Windows 10 WDK and Windows 10 ADK **builds 20236 and later** require specifying the digest algorithm. The SignTool `sign` command requires the `/fd` **file digest algorithm** and the `/td` **timestamp digest algorithm** option to be specified during signing and timestamping, respectively. An error (error code 1) will be thrown if `/fd` is not specified during signing and if `/td` is not specified during timestamping.

At the command prompt, type the following:

Syntax

Console

```
signtool [command] [options] [file_name | ...]
```

Parameters

Argument	Description
<code>command</code>	One of four commands (<code>catdb</code> , <code>sign</code> , <code>Timestamp</code> , or <code>Verify</code>) that specifies an operation to perform on a file. For a description of each command, see the next table.
<code>options</code>	An option that modifies a command. In addition to the global <code>/q</code> and <code>/v</code> options, each command supports a unique set of options.
<code>file_name</code>	The path to a file to sign.

The following commands are supported by Sign Tool. Each command is used with distinct sets of options, which are listed in their respective sections.

Command	Description
<code>catdb</code>	Adds a catalog file to, or removes it from, a catalog database. Catalog databases are used for automatic lookup of catalog files and are identified by GUID. For a list of the options supported by the <code>catdb</code> command, see catdb Command Options .
<code>sign</code>	Digitally signs files. Digital signatures protect files from tampering, and enable users to verify the signer based on a signing certificate. For a list of the options supported by the <code>sign</code> command, see sign Command Options .
<code>TimeStamp</code>	Time-stamps files. For a list of the options supported by the <code>TimeStamp</code> command, see TimeStamp Command Options .
<code>Verify</code>	Verifies the digital signature of files by determining whether the signing certificate was issued by a trusted authority, whether the signing certificate has been revoked, and, optionally, whether the signing certificate is valid for a specific policy. For a list of the options supported by the <code>verify</code> command, see Verify Command Options .

The following options apply to all Sign Tool commands.

Global option	Description
<code>/q</code>	Displays no output if the command runs successfully, and displays minimal output if the command fails.
<code>/v</code>	Displays verbose output regardless of whether the command runs successfully or fails, and displays warning messages.
<code>/debug</code>	Displays debugging information.

catdb Command Options

The following table lists the options that can be used with the `catdb` command.

Catdb option	Description
<code>/d</code>	Specifies that the default catalog database is updated. If neither the <code>/d</code> nor the <code>/g</code> option is used, Sign Tool updates the system component and driver database.
<code>/g GUID</code>	Specifies that the catalog database identified by the globally unique identifier <i>GUID</i> is updated.

Catdb	Description
/r	Removes the specified catalogs from the catalog database. If this option is not specified, Sign Tool adds the specified catalogs to the catalog database.
/u	Specifies that a unique name is automatically generated for the added catalog files. If necessary, the catalog files are renamed to prevent name conflicts with existing catalog files. If this option is not specified, Sign Tool overwrites any existing catalog that has the same name as the catalog being added.

sign Command Options

The following table lists the options that can be used with the `sign` command.

Sign command	Description
/a	Automatically selects the best signing certificate. Sign Tool will find all valid certificates that satisfy all specified conditions and select the one that is valid for the longest time. If this option is not present, Sign Tool expects to find only one valid signing certificate.
/ac <i>file</i>	Adds an additional certificate from <i>file</i> to the signature block.
/as	Appends this signature. If no primary signature is present, this signature is made the primary signature instead.
/c <i>CertTemplateName</i>	Specifies the Certificate Template Name (a Microsoft extension) for the signing certificate.
/csp <i>CSPName</i>	Specifies the cryptographic service provider (CSP) that contains the private key container.
/d <i>Desc</i>	Specifies a description of the signed content.
/du <i>URL</i>	Specifies a Uniform Resource Locator (URL) for the expanded description of the signed content.
/f <i>SignCertFile</i>	Specifies the signing certificate in a file. If the file is in Personal Information Exchange (PFX) format and protected by a password, use the /p option to specify the password. If the file does not contain private keys, use the /csp and /kc options to specify the CSP and private key container name.
/fd	Specifies the file digest algorithm to use for creating file signatures. Note: An error is generated if the /fd switch is not provided while signing.

Sign command option	Description
<code>/fd certHash</code>	<p>Specifying the string <i>certHash</i> will default to the algorithm used on the signing certificate.</p> <p>Note: An error is generated if the <code>/fd</code> switch is not provided while signing.</p>
<code>/i IssuerName</code>	<p>Specifies the name of the issuer of the signing certificate. This value can be a substring of the entire issuer name.</p>
<code>/kc PrivKeyContainerName</code>	<p>Specifies the private key container name.</p>
<code>/n SubjectName</code>	<p>Specifies the name of the subject of the signing certificate. This value can be a substring of the entire subject name.</p>
<code>/nph</code>	<p>If supported, suppresses page hashes for executable files. The default is determined by the SIGNTOOL_PAGE_HASHES environment variable and by the wintrust.dll version. This option is ignored for non-PE files.</p>
<code>/p Password</code>	<p>Specifies the password to use when opening a PFX file. (Use the <code>/f</code> option to specify a PFX file.)</p>
<code>/p7 Path</code>	<p>Specifies that a Public Key Cryptography Standards (PKCS) #7 file is produced for each specified content file. PKCS #7 files are named <i>path\filename.p7</i>.</p>
<code>/p7ce Value</code>	<p>Specifies options for the signed PKCS #7 content. Set <i>Value</i> to "Embedded" to embed the signed content in the PKCS #7 file, or to "DetachedSignedData" to produce the signed data portion of a detached PKCS #7 file. If the <code>/p7ce</code> option is not used, the signed content is embedded by default.</p>
<code>/p7co <OID></code>	<p>Specifies the object identifier (OID) that identifies the signed PKCS #7 content.</p>
<code>/ph</code>	<p>If supported, generates page hashes for executable files.</p>
<code>/r RootSubjectName</code>	<p>Specifies the name of the subject of the root certificate that the signing certificate must chain to. This value may be a substring of the entire subject name of the root certificate.</p>
<code>/s StoreName</code>	<p>Specifies the store to open when searching for the certificate. If this option is not specified, the <code>My</code> store is opened.</p>
<code>/sha1 Hash</code>	<p>Specifies the SHA1 hash of the signing certificate. The SHA1 hash is commonly specified when multiple certificates satisfy the criteria specified by the remaining switches.</p>
<code>/sm</code>	<p>Specifies that a machine store, instead of a user store, is used.</p>

Sign command option	Description
/t URL	Specifies the URL of the time stamp server. If this option (or /tr) is not present, the signed file will not be time stamped. A warning is generated if time stamping fails. This option cannot be used with the /tr option.
/td alg	Used with the /tr option to request a digest algorithm used by the RFC 3161 time stamp server. Note: An error is generated if /td is not provided while timestamping.
/tr URL	Specifies the URL of the RFC 3161 time stamp server. If this option (or /t) is not present, the signed file will not be time stamped. A warning is generated if time stamping fails. This option cannot be used with the /t option.
/u Usage	Specifies the enhanced key usage (EKU) that must be present in the signing certificate. The usage value can be specified by OID or string. The default usage is "Code Signing" (1.3.6.1.5.5.7.3.3).
/uw	Specifies usage of "Windows System Component Verification" (1.3.6.1.4.1.311.10.3.6).

For usage examples, see [Using SignTool to Sign a File](#).

TimeStamp Command Options

The following table lists the options that can be used with the `TimeStamp` command.

TimeStamp option	Description
/p7	Time stamps PKCS #7 files.
/t URL	Specifies the URL of the time stamp server. The file being time stamped must have previously been signed. Either the /t or the /tr option is required.
/td alg	Used with the /tr option to request a digest algorithm used by the RFC 3161 time stamp server. Note: An error is generated if /td is not provided while timestamping.
/tp index	Time stamps the signature at <i>index</i> .
/tr URL	Specifies the URL of the RFC 3161 time stamp server. The file being time stamped must have previously been signed. Either the /tr or the /t option is required.

For a usage example, see [Adding Time Stamps to Previously Signed Files](#).

Verify Command Options

Verify option	Description
/a	Specifies that all methods can be used to verify the file. First, the catalog databases are searched to determine whether the file is signed in a catalog. If the file is not signed in any catalog, Sign Tool attempts to verify the file's embedded signature. This option is recommended when verifying files that may or may not be signed in a catalog. Examples of these files include Windows files or drivers.
/ad	Finds the catalog by using the default catalog database.
/ag <i>CatDBGUID</i>	Finds the catalog in the catalog database that is identified by the <i>CatDBGUID</i> .
/all	Verifies all signatures in a file that includes multiple signatures.
/as	Finds the catalog by using the system component (driver) catalog database.
/c <i>CatFile</i>	Specifies the catalog file by name.
/d	Specifies that Sign Tool should print the description and the description URL.
/ds <i>Index</i>	Verifies the signature at a specified position.
/hash (SHA1 SHA256)	Specifies an optional hash algorithm to use when searching for a file in a catalog.
/kp	Specifies that verification should be performed with the kernel-mode driver signing policy.
/ms	Uses multiple verification semantics. This is the default behavior of a WinVerifyTrust call on Windows 8 and above.
/o <i>Version</i>	Verifies the file by operating system version. <i>Version</i> has the following form: <i>PlatformID:VerMajor.VerMinor.BuildNumber</i> . <i>PlatformID</i> represents the underlying value of a PlatformID enumeration member. Important: The use of the /o switch is recommended. If /o is not specified, SignTool.exe may return unexpected results. For example, if you do not include the /o switch, system catalogs that validate correctly on an older operating system may not validate correctly on a newer operating system.
/p7	Verifies PKCS #7 files. No existing policies are used for PKCS #7 validation. The signature is checked and a chain is built for the signing certificate.
/pa	Specifies that the Default Authenticode Verification Policy should be used. If the /pa option is not specified, Sign Tool uses the Windows Driver Verification Policy. This option cannot be used with the catdb options.

Verify option	Description
/pg <i>PolicyGUID</i>	Specifies a verification policy by GUID. The <i>PolicyGUID</i> corresponds to the ActionID of the verification policy. This option cannot be used with the <code>catdb</code> options.
/ph	Specifies that Sign Tool should print and verify page hash values.
/r <i>RootSubjectName</i>	Specifies the name of the subject of the root certificate that the signing certificate must chain to. This value can be a substring of the entire subject name of the root certificate.
/tw	Specifies that a warning should be generated if the signature is not time stamped.

For usage examples, see [Using SignTool to Verify a File Signature](#).

Return Value

Sign Tool returns one of the following exit codes when it terminates.

Exit code	Description
0	Execution was successful.
1	Execution has failed.
2	Execution has completed with warnings.

Examples

The following command adds the catalog file MyCatalogFileName.cat to the system component and driver database. The /u option generates a unique name if necessary to prevent replacing an existing catalog file named `MyCatalogFileName.cat`.

Console

```
signtool catdb /v /u MyCatalogFileName.cat
```

The following command signs a file automatically by using the best certificate.

Console

```
signtool sign /a /fd SHA256 MyFile.exe
```

The following command digitally signs a file by using a certificate stored in a password-protected PFX file.

Console

```
signtool sign /f MyCert.pfx /p MyPassword /fd SHA256 MyFile.exe
```

The following command digitally signs and time-stamps a file. The certificate used to sign the file is stored in a PFX file.

Console

```
signtool sign /f MyCert.pfx /t http://timestamp.digicert.com /fd SHA256 MyFile.exe
```

The following command signs a file by using a certificate located in the `My` store that has a subject name of `My Company Certificate`.

Console

```
signtool sign /n "My Company Certificate" /fd SHA256 MyFile.exe
```

The following command signs an ActiveX control and provides information that's displayed in the browser when the user is prompted to install the control.

Console

```
Signtool sign /f MyCert.pfx /d: "MyControl" /du http://www.example.com/MyControl/info.html /fd SHA256 MyControl.exe
```

The following command time-stamps a file that has already been digitally signed.

Console

```
signtool timestamp /t http://timestamp.digicert.com MyFile.exe
```

The following command time-stamps a file using an RFC 3161 timestamp server.

Console

```
signtool timestamp /tr http://timestamp.digicert.com /td SHA256 MyFile.exe
```

The following command verifies that a file has been signed.

Console

```
signtool verify MyFile.exe
```

The following command verifies a system file that may be signed in a catalog.

Console

```
signtool verify /a SystemFile.dll
```

The following command verifies a system file that is signed in a catalog named [MyCatalog.cat](#).

Console

```
signtool verify /c MyCatalog.cat SystemFile.dll
```

See also

- [Tools](#)
- [Developer command-line shells](#)

Sn.exe (Strong Name Tool)

Article • 03/18/2022

The Strong Name tool (Sn.exe) helps sign assemblies with [strong names](#). Sn.exe provides options for key management, signature generation, and signature verification.

⚠ Warning

Do not rely on strong names for security. They provide a unique identity only.

For more information on strong naming and strong-named assemblies, see [Strong-Named Assemblies](#) and [How to: Sign an Assembly with a Strong Name](#).

The Strong Name tool is automatically installed with Visual Studio. To start the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

ⓘ Note

On 64-bit computers, run the 32-bit version of Sn.exe by using the Developer Command Prompt for Visual Studio and the 64-bit version by using the Visual Studio x64 Win64 Command Prompt.

At the command prompt, type the following:

Syntax

Console

```
sn [-quiet][option [parameter(s)]]
```

Parameters

[] [Expand table](#)

Option	Description
<code>-a identityKeyPairFile signaturePublicKeyFile</code>	Generates AssemblySignatureKeyAttribute data to migrate the identity key to the signature key from a file.

Option	Description
<code>-ac identityPublicKeyFile identityKeyPairContainer signaturePublicKeyFile</code>	Generates AssemblySignatureKeyAttribute data to migrate the identity key to the signature key from a key container.
<code>-c [csp]</code>	Sets the default cryptographic service provider (CSP) to use for strong name signing. This setting applies to the entire computer. If you do not specify a CSP name, Sn.exe clears the current setting.
<code>-d container</code>	Deletes the specified key container from the strong name CSP.
<code>-D assembly1 assembly2</code>	Verifies that two assemblies differ only by signature. This is often used as a check after an assembly has been re-signed with a different key pair.
<code>-e assembly outfile</code>	Extracts the public key from <i>assembly</i> and stores it in <i>outfile</i> .
<code>-h</code>	Displays command syntax and options for the tool.
<code>-i infile container</code>	Installs the key pair from <i>infile</i> in the specified key container. The key container resides in the strong name CSP.
<code>-k [keysize] outfile</code>	Generates a new RSACryptoServiceProvider key of the specified size and writes it to the specified file. Both a public and private key are written to the file.
	<p>If you do not specify a key size, a 1,024-bit key is generated by default if you have the Microsoft enhanced cryptographic provider installed; otherwise, a 512-bit key is generated.</p> <p>The <i>keysize</i> parameter supports key lengths from 384 bits to 16,384 bits in increments of 8 bits if you have the Microsoft enhanced cryptographic provider installed. It supports key lengths from 384 bits to 512 bits in increments of 8 bits if you have the Microsoft base cryptographic provider installed.</p>
<code>-m [y or n]</code>	<p>Specifies whether key containers are computer-specific, or user-specific. If you specify <i>y</i>, key containers are computer-specific. If you specify <i>n</i>, key containers are user-specific.</p> <p>If neither <i>y</i> nor <i>n</i> is specified, this option displays the current setting.</p>
<code>-o infile [outfile]</code>	Extracts the public key from the <i>infile</i> and stores it in a .csv file. A comma separates each byte of the public key. This format is useful for hard-coding references to keys as initialized arrays in source code. If you do not specify an <i>outfile</i> , this option places the output on the Clipboard. Note: This option does not verify that the input is

Option	Description
	only a public key. If the <code>infile</code> contains a key pair with a private key, the private key is also extracted.
<code>-p infile outfile [hashalg]</code>	Extracts the public key from the key pair in <code>infile</code> and stores it in <code>outfile</code> , optionally using the RSA algorithm specified by <code>hashalg</code> . This public key can be used to delay-sign an assembly using the <code>/delaySign+</code> and <code>/keyfile</code> options of the Assembly Linker (Al.exe) . When an assembly is delay-signed, only the public key is set at compile time and space is reserved in the file for the signature to be added later, when the private key is known.
<code>-pc container outfile [hashalg]</code>	Extracts the public key from the key pair in <code>container</code> and stores it in <code>outfile</code> . If you use the <code>hashalg</code> option, the RSA algorithm is used to extract the public key.
<code>-Pb [y or n]</code>	Specifies whether the strong-name bypass policy is enforced. If you specify <code>y</code> , strong names for full-trust assemblies are not validated when loaded into a full-trust AppDomain . If you specify <code>n</code> , strong names are validated for correctness, but not for a specific strong name. The StrongNameIdentityPermission has no effect on full-trust assemblies. You must perform your own check for a strong name match.
	If neither <code>y</code> nor <code>n</code> is specified, this option displays the current setting. The default is <code>y</code> . Note: On 64-bit computers, you must set this parameter in both the 32-bit and the 64-bit instances of Sn.exe.
<code>-q[uiet]</code>	Specifies quiet mode; suppresses the display of success messages.
<code>-R[a] assembly infile</code>	Re-signs a previously signed or delay-signed assembly with the key pair in <code>infile</code> .
	If <code>-Ra</code> is used, hashes are recomputed for all files in the assembly.
<code>-Rc[a] assembly container</code>	Re-signs a previously signed or delay-signed assembly with the key pair in <code>container</code> .
	If <code>-Rca</code> is used, hashes are recomputed for all files in the assembly.
<code>-Rh assembly</code>	Recomputes hashes for all files in the assembly.
<code>-t[p] infile</code>	Displays the token for the public key stored in <code>infile</code> . The contents of <code>infile</code> must be a public key previously generated from a key pair file using <code>-p</code> . Do not use the <code>-t[p]</code> option to extract the token directly from a key pair file.
	Sn.exe computes the token by using a hash function from the public key. To save space, the common language runtime stores

Option	Description
	<p>public key tokens in the manifest as part of a reference to another assembly when it records a dependency to an assembly that has a strong name. The -tp option displays the public key in addition to the token. If the AssemblySignatureKeyAttribute attribute has been applied to the assembly, the token is for the identity key, and the name of the hash algorithm and the identity key is displayed.</p> <p>Note that this option does not verify the assembly signature and should not be used to make trust decisions. This option only displays the raw public key token data.</p>
-T[tp] assembly	<p>Displays the public key token for <i>assembly</i>. The <i>assembly</i> must be the name of a file that contains an assembly manifest.</p> <p>Sn.exe computes the token by using a hash function from the public key. To save space, the runtime stores public key tokens in the manifest as part of a reference to another assembly when it records a dependency to an assembly that has a strong name. The -Tp option displays the public key in addition to the token. If the AssemblySignatureKeyAttribute attribute has been applied to the assembly, the token is for the identity key, and the name of the hash algorithm and the identity key is displayed.</p> <p>Note that this option does not verify the assembly signature and should not be used to make trust decisions. This option only displays the raw public key token data.</p>
-TS assembly infile	<p>Test-signs the signed or partially signed <i>assembly</i> with the key pair in <i>infile</i>.</p>
-TSc assembly container	<p>Test-signs the signed or partially signed <i>assembly</i> with the key pair in the key container <i>container</i>.</p>
-v assembly	<p>Verifies the strong name in <i>assembly</i>, where <i>assembly</i> is the name of a file that contains an assembly manifest.</p>
-vf assembly	<p>Verifies the strong name in <i>assembly</i>. Unlike the -v option, -vf forces verification even if it is disabled using the -Vr option.</p>
-Vk regfile.reg assembly [userlist] [infile]	<p>Creates a registration entries (.reg) file you can use to register the specified assembly for verification skipping. The rules for assembly naming that apply to the -Vr option apply to -Vk as well. For information about the <i>userlist</i> and <i>infile</i> options, see the -Vr option.</p>
-V1	<p>Lists current settings for strong-name verification on this computer.</p>
-Vr assembly [userlist] [infile]	<p>Registers <i>assembly</i> for verification skipping. Optionally, you can specify a comma-separated list of user names the skip verification should apply to. If you specify <i>infile</i>, verification remains enabled,</p>

Option	Description
	<p>but the public key in <i>infile</i> is used in verification operations. You can specify <i>assembly</i> in the form <i>*;strongname</i> to register all assemblies with the specified strong name. For <i>strongname</i>, specify the string of hexadecimal digits representing the tokenized form of the public key. See the -t and -T options to display the public key token. Caution: Use this option only during development. Adding an assembly to the skip verification list creates a security vulnerability. A malicious assembly could use the fully specified assembly name (assembly name, version, culture, and public key token) of the assembly added to the skip verification list to fake its identity. This would allow the malicious assembly to also skip verification.</p>
-Vu <i>assembly</i>	<p>Unregisters <i>assembly</i> for verification skipping. The same rules for assembly naming that apply to -Vr apply to -Vu.</p>
-Vx	<p>Removes all verification-skipping entries.</p>
-?	<p>Displays command syntax and options for the tool.</p>

① Note

All Sn.exe options are case-sensitive and must be typed exactly as shown to be recognized by the tool.

Remarks

The **-R** and **-Rc** options are useful with assemblies that have been delay-signed. In this scenario, only the public key has been set at compile time and signing is performed later, when the private key is known.

① Note

For parameters (for example, **-Vr**) that write to protected resources such as the registry, run SN.exe as an administrator.

The Strong Name tool assumes that public/private key pairs are generated with the **AT_SIGNATURE** algorithm identifier. Public/private key pairs generated with the **AT_KEYEXCHANGE** algorithm generate an error.

Examples

The following command creates a new, random key pair and stores it in `keyPair.snk`.

Console

```
sn -k keyPair.snk
```

The following command stores the key in `keyPair.snk` in the container `MyContainer` in the strong name CSP.

Console

```
sn -i keyPair.snk MyContainer
```

The following command extracts the public key from `keyPair.snk` and stores it in `publicKey.snk`.

Console

```
sn -p keyPair.snk publicKey.snk
```

The following command displays the public key and the token for the public key contained in `publicKey.snk`.

Console

```
sn -tp publicKey.snk
```

The following command verifies the assembly `MyAsm.dll`.

Console

```
sn -v MyAsm.dll
```

The following command deletes `MyContainer` from the default CSP.

Console

```
sn -d MyContainer
```

See also

- [Tools](#)
- [Al.exe \(Assembly Linker\)](#)
- [Strong-Named Assemblies](#)
- [Developer command-line shells](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SOS.dll (SOS debugging extension)

Article • 07/20/2022

ⓘ Important

This article concerns the .NET Framework version of the SOS debugging extension. For information about the newer .NET (Core) version of the tool, see [SOS debugging extension](#).

The SOS debugging extension (SOS.dll) helps you debug managed programs in Visual Studio and in the Windows debugger (WinDbg.exe) by providing information about the internal Common Language Runtime (CLR) environment. This tool requires your project to have unmanaged debugging enabled. SOS.dll is automatically installed with .NET Framework. To use SOS.dll in Visual Studio, install the [Windows Driver Kit \(WDK\)](#).

Syntax

Console

```
![command] [options]
```

Commands

[+] Expand table

Command	Description
AnalyzeOOM (ao)	Displays the information for the last out of memory (OOM) that occurred on an allocation request to the garbage collection heap. (In server garbage collection, it displays OOM, if any, on each garbage collection heap.)
BPMD [-nofuturemodule] <module name> <method name>] [-md <MethodDesc>] -list -clear <pending breakpoint number> -clearall	Creates a breakpoint at the specified method in the specified module. If the specified module and method have not been loaded, this command waits for a notification that the module was loaded and just-in-time (JIT) compiled before creating a breakpoint. You can manage the list of pending breakpoints by using the -list , -clear , and -clearall options:

Command	Description
	<p>The -list option generates a list of all the pending breakpoints. If a pending breakpoint has a non-zero module ID, that breakpoint is specific to a function in that particular loaded module. If the pending breakpoint has a zero module ID, that breakpoint applies to modules that have not yet been loaded.</p> <p>Use the -clear or -clearall option to remove pending breakpoints from the list.</p>
CLRStack [-a] [-l] [-p] [-n]	<p>Provides a stack trace of managed code only.</p> <p>The -p option shows arguments to the managed function.</p> <p>The -l option shows information on local variables in a frame. The SOS Debugging Extension cannot retrieve local names, so the output for local names is in the format <i><local address></i> = <i><value></i>.</p> <p>The -a(all) option is a shortcut for -l and -p combined.</p> <p>The -n option disables the display of source file names and line numbers. If the debugger has the option SYMOPT_LOAD_LINES specified, SOS will look up the symbols for every managed frame and if successful will display the corresponding source file name and line number. The -n (No line numbers) parameter can be specified to disable this behavior.</p> <p>The SOS Debugging Extension does not display transition frames on x64 and IA-64-based platforms.</p>
COMState	<p>Lists the COM apartment model for each thread and a Context pointer, if available.</p>
DumpArray [-start <startIndex>] [-length <length>] [-details] [-nofields] <array object address> -or- DA [-start <startIndex>] [-length <length>] [-detail] [-nofields] <array object address>	<p>Examines elements of an array object.</p> <p>The -start option specifies the starting index at which to display elements.</p> <p>The -length option specifies how many elements to show.</p> <p>The -details option displays details of the element using the DumpObj and DumpVC formats.</p> <p>The -nofields option prevents arrays from displaying. This option is available only when the -detail option is specified.</p>
DumpAssembly <assembly address>	<p>Displays information about an assembly.</p> <p>The DumpAssembly command lists multiple modules, if they exist.</p>

Command	Description
	<p>You can get an assembly address by using the DumpDomain command.</p>
DumpClass < <i>EEClass address</i> >	<p>Displays information about the <code>EEClass</code> structure associated with a type.</p> <p>The DumpClass command displays static field values but does not display nonstatic field values.</p> <p>Use the DumpMT, DumpObj, Name2EE, or Token2EE command to get an <code>EEClass</code> structure address.</p>
DumpDomain [< <i>domain address</i> >]	<p>Enumerates each Assembly object that is loaded within the specified AppDomain object address. When called with no parameters, the DumpDomain command lists all AppDomain objects in a process.</p>
DumpHeap [-stat] [-strings] [-short] [-min < <i>size</i> >] [-max < <i>size</i> >] [-thinlock] [-startAtLowerBound] [-mt < <i>MethodTable address</i> >] [-type < <i>partial type name</i> >] [<i>start [end]</i>]	<p>Displays information about the garbage-collected heap and collection statistics about objects.</p> <p>The DumpHeap command displays a warning if it detects excessive fragmentation in the garbage collector heap.</p> <p>The -stat option restricts the output to the statistical type summary.</p> <p>The -strings option restricts the output to a statistical string value summary.</p> <p>The -short option limits output to just the address of each object. This lets you easily pipe output from the command to another debugger command for automation.</p> <p>The -min option ignores objects that are less than the <code>size</code> parameter, specified in bytes (hex).</p> <p>The -max option ignores objects that are larger than the <code>size</code> parameter, specified in bytes (hex).</p> <p>The -thinlock option reports ThinLocks. For more information, see the SyncBlk command.</p>
	<p>The <code>-startAtLowerBound</code> option forces the heap walk to begin at the lower bound of a supplied address range. During the planning phase, the heap is often not walkable because objects are being moved. This option forces DumpHeap to begin its walk at the specified lower bound. You must supply the address of a valid object as the lower bound for this option to work. You can display memory at the address of a bad object to manually find the next method table. If the garbage</p>

Command	Description
	<p>collection is currently in a call to <code>memcpy</code>, you may also be able to find the address of the next object by adding the size to the start address, which is supplied as a parameter.</p> <p>The -mt option lists only those objects that correspond to the specified <code>MethodTable</code> structure.</p> <p>The -type option lists only those objects whose type name is a substring match of the specified string.</p> <p>The <code>start</code> parameter begins listing from the specified address.</p> <p>The <code>end</code> parameter stops listing at the specified address.</p>
DumpIL < <i>Managed DynamicMethod object</i> > < <i>DynamicMethodDesc pointer</i> > < <i>MethodDesc pointer</i> >	<p>Displays the common intermediate language (CIL) that is associated with a managed method.</p> <p>Note that dynamic CIL is emitted differently than CIL that is loaded from an assembly. Dynamic CIL refers to objects in a managed object array rather than to metadata tokens.</p>
DumpLog [-addr < <i>addressOfStressLog</i> >] [< <i>filename</i> >]	<p>Writes the contents of an in-memory stress log to the specified file. If you do not specify a name, this command creates a file called <code>StressLog.txt</code> in the current directory.</p> <p>The in-memory stress log helps you diagnose stress failures without using locks or I/O. To enable the stress log, set the following registry keys under <code>HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NETFramework:</code></p> <p>(DWORD) <code>StressLog</code> = 1</p> <p>(DWORD) <code>LogFacility</code> = 0xffffffff</p> <p>(DWORD) <code>StressLogSize</code> = 65536</p> <p>The optional <code>-addr</code> option lets you specify a stress log other than the default log.</p>
DumpMD < <i>MethodDesc address</i> >	<p>Displays information about a <code>MethodDesc</code> structure at the specified address.</p> <p>You can use the IP2MD command to get the <code>MethodDesc</code> structure address from a managed function.</p>
DumpMT [-MD] < <i>MethodTable address</i> >	<p>Displays information about a method table at the specified address. Specifying the -MD option displays a list of all methods defined with the object.</p>

Command	Description
	Each managed object contains a method table pointer.
DumpMethodSig <i><sigaddr></i> <i><moduleaddr></i>	Displays information about a <code>MethodSig</code> structure at the specified address.
DumpModule [-mt] <i><Module address></i>	Displays information about a module at the specified address. The <code>-mt</code> option displays the types defined in a module and the types referenced by the module You can use the DumpDomain or DumpAssembly command to retrieve a module's address.
DumpObj [-nofields] <i><object address></i> -or- DO <i><object address></i>	Displays information about an object at the specified address. The DumpObj command displays the fields, the <code>EEClass</code> structure information, the method table, and the size of the object. You can use the DumpStackObjects command to retrieve an object's address. Note that you can run the DumpObj command on fields of type <code>CLASS</code> because they are also objects.
	The <code>-nofields</code> option prevents fields of the object being displayed, it is useful for objects like String.
DumpRuntimeTypes	Displays the runtime type objects in the garbage collector heap and lists their associated type names and method tables.
DumpStack [-EE] [-n] [<i>top stack</i> [<i>bottom stack</i>]]	Displays a stack trace. The <code>-EE</code> option causes the DumpStack command to display only managed functions. Use the <code>top</code> and <code>bottom</code> parameters to limit the stack frames displayed on x86 platforms. The <code>-n</code> option disables the display of source file names and line numbers. If the debugger has the option <code>SYMOPT_LOAD_LINES</code> specified, SOS will look up the symbols for every managed frame and if successful will display the corresponding source file name and line number. The <code>-n</code> (No line numbers) parameter can be specified to disable this behavior. On x86 and x64 platforms, the DumpStack command creates a verbose stack trace.
	On IA-64-based platforms, the DumpStack command mimics the

Command	Description
	debugger's K command. The <code>top</code> and <code>bottom</code> parameters are ignored on IA-64-based platforms.
<code>DumpSig <sigaddr> <moduleaddr></code>	Displays information about a <code>Sig</code> structure at the specified address.
<code>DumpSigElem <sigaddr> <moduleaddr></code>	Displays a single element of a signature object. In most cases, you should use <code>DumpSig</code> to look at individual signature objects. However, if a signature has been corrupted in some way, you can use <code>DumpSigElem</code> to read the valid portions of it.
<code>DumpStackObjects [-verify] [top stack [bottom stack]]</code> -or- <code>DSO [-verify] [top stack [bottom stack]]</code>	<p>Displays all managed objects found within the bounds of the current stack.</p> <p>The <code>-verify</code> option validates each non-static <code>CLASS</code> field of an object field.</p> <p>Use the <code>DumpStackObject</code> command with stack tracing commands such as the K command and the <code>CLRStack</code> command to determine the values of local variables and parameters.</p>
<code>DumpVC <MethodTable address> <Address></code>	<p>Displays information about the fields of a value class at the specified address.</p> <p>The <code>MethodTable</code> parameter allows the <code>DumpVC</code> command to correctly interpret fields. Value classes do not have a method table as their first field.</p>
<code>EEHeap [-gc] [-loader]</code>	<p>Displays information about process memory consumed by internal CLR data structures.</p> <p>The <code>-gc</code> and <code>-loader</code> options limit the output of this command to garbage collector or loader data structures.</p> <p>The information for the garbage collector lists the ranges of each segment in the managed heap. If the pointer falls within a segment range given by <code>-gc</code>, the pointer is an object pointer.</p>
<code>EEStack [-short] [-EE]</code>	<p>Runs the <code>DumpStack</code> command on all threads in the process.</p> <p>The <code>-EE</code> option is passed directly to the <code>DumpStack</code> command. The <code>-short</code> parameter limits the output to the following kinds of threads:</p> <ul style="list-style-type: none"> Threads that have taken a lock. Threads that have been stalled in order to allow a garbage collection. Threads that are currently in managed code.

Command	Description
EEVersion	Displays the CLR version.
EHInfo [<MethodDesc address>] [<Code address>]	Displays the exception handling blocks in a specified method. This command displays the code addresses and offsets for the clause block (the <code>try</code> block) and the handler block (the <code>catch</code> block).
FAQ	Displays frequently asked questions.
FinalizeQueue [-detail] [-allReady] [-short]	<p>Displays all objects registered for finalization.</p> <p>The <code>-detail</code> option displays extra information about any <code>SyncBlocks</code> that need to be cleaned up, and any <code>RuntimeCallableWrapperWrappers</code> (RCWs) that await cleanup. Both of these data structures are cached and cleaned up by the finalizer thread when it runs.</p> <p>The <code>-allReady</code> option displays all objects that are ready for finalization, regardless of whether they are already marked by the garbage collection as such, or will be marked by the next garbage collection. The objects that are in the "ready for finalization" list are finalizable objects that are no longer rooted. This option can be very expensive, because it verifies whether all the objects in the finalizable queues are still rooted.</p> <p>The <code>-short</code> option limits the output to the address of each object. If it is used in conjunction with <code>-allReady</code>, it enumerates all objects that have a finalizer that are no longer rooted. If it is used independently, it lists all objects in the finalizable and "ready for finalization" queues.</p>
FindAppDomain <Object address>	Determines the application domain of an object at the specified address.
FindRoots -gen <N> -gen any <object address>	Causes the debugger to break in the debugger on the next collection of the specified generation. The effect is reset as soon as the break occurs. To break on the next collection, you have to reissue the command. The <code><object address></code> form of this command is used after the break caused by the <code>-gen</code> or <code>-gen any</code> has occurred. At that time, the debugger is in the right state for FindRoots to identify roots for objects from the current condemned generations.
GCHandle [-perdomain]	<p>Displays statistics about garbage collector handles in the process.</p> <p>The <code>-perdomain</code> option arranges the statistics by application domain.</p>
	Use the GCHandle command to find memory leaks caused by garbage collector handle leaks. For example, a memory leak occurs when code retains a large array because a strong garbage collector handle still points to it, and the handle is discarded without freeing it.

Command	Description
GCHandleLeaks	Searches memory for any references to strong and pinned garbage collector handles in the process and displays the results. If a handle is found, the GCHandleLeaks command displays the address of the reference. If a handle is not found in memory, this command displays a notification.
GCInfo < <i>MethodDesc address</i> >< <i>Code address</i> >	Displays data that indicates when registers or stack locations contain managed objects. If a garbage collection occurs, the collector must know the locations of references to objects so it can update them with new object pointer values.
GCRoot [-nostacks] < <i>Object address</i> >	<p>Displays information about references (or roots) to an object at the specified address.</p> <p>The GCRoot command examines the entire managed heap and the handle table for handles within other objects and handles on the stack. Each stack is then searched for pointers to objects, and the finalizer queue is also searched.</p>
	<p>This command does not determine whether a stack root is valid or is discarded. Use the CLRStack and U commands to disassemble the frame that the local or argument value belongs to in order to determine if the stack root is still in use.</p> <p>The -nostacks option restricts the search to garbage collector handles and reachable objects.</p>
GCWhere < <i>object address</i> >	Displays the location and size in the garbage collection heap of the argument passed in. When the argument lies in the managed heap but is not a valid object address, the size is displayed as 0 (zero).
help [< <i>command</i> >] [faq]	<p>Displays all available commands when no parameter is specified, or displays detailed help information about the specified command.</p> <p>The faq parameter displays answers to frequently asked questions.</p>
HeapStat [-inclUnrooted -iu]	Displays the generation sizes for each heap and the total free space in each generation on each heap. If the -inclUnrooted option is specified, the report includes information about the managed objects from the garbage collection heap that is no longer rooted.
HistClear	<p>Releases any resources used by the family of Hist commands.</p> <p>Generally, you do not have to explicitly call HistClear, because each HistInit cleans up the previous resources.</p>
HistInit	Initializes the SOS structures from the stress log saved in the debuggee.

Command	Description
HistObj <obj_address>	Examines all stress log relocation records and displays the chain of garbage collection relocations that may have led to the address passed in as an argument.
HistObjFind <obj_address>	Displays all the log entries that reference an object at the specified address.
HistRoot <root>	Displays information related to both promotions and relocations of the specified root. The root value can be used to track the movement of an object through the garbage collections.
IP2MD <Code address>	Displays the <code>MethodDesc</code> structure at the specified address in code that has been JIT-compiled.
ListNearObj (1no) <obj_address>	Displays the objects preceding and following the specified address. The command looks for the address in the garbage collection heap that looks like a valid beginning of a managed object (based on a valid method table) and the object following the argument address.
MinidumpMode [0] [1]	Prevents running unsafe commands when using a minidump. Pass 0 to disable this feature or 1 to enable this feature. By default, the MinidumpMode value is set to 0. Minidumps created with the <code>.dump /m</code> command or <code>.dump</code> command have limited CLR-specific data and allow you to run only a subset of SOS commands correctly. Some commands may fail with unexpected errors because required areas of memory are not mapped or are only partially mapped. This option protects you from running unsafe commands against minidumps.
Name2EE <module name> <type or method name>	Displays the <code>MethodTable</code> structure and <code>EEClass</code> structure for the specified type or method in the specified module. The specified module must be loaded in the process. -or-
Name2EE <module name>!<type or method name>	To get the proper type name, browse the module by using the Ildasm.exe (IL Disassembler) . You can also pass * as the module name parameter to search all loaded managed modules. The <i>module name</i> parameter can also be the debugger's name for a module, such as <code>mscorlib</code> or <code>image00400000</code> .
	This command supports the Windows debugger syntax of < <code>module</code> > ! < <code>type</code> >. The type must be fully qualified.

Command	Description
ObjSize [<Object address>] [-aggregate] [-stat]	<p>Displays the size of the specified object. If you do not specify any parameters, the ObjSize command displays the size of all objects found on managed threads, displays all garbage collector handles in the process, and totals the size of any objects pointed to by those handles. The ObjSize command includes the size of all child objects in addition to the parent.</p>
	<p>The -aggregate option can be used in conjunction with the -stat argument to get a detailed view of the types that are still rooted. By using !dumpheap -stat and !objsize -aggregate -stat, you can determine which objects are no longer rooted and diagnose various memory issues.</p>
PrintException [-nested] [-lines] [<Exception object address>]	<p>Displays and formats fields of any object derived from the Exception class at the specified address. If you do not specify an address, the PrintException command displays the last exception thrown on the current thread.</p>
-or-	<p>The -nested option displays details about nested exception objects.</p>
PE [-nested] [<Exception object address>]	<p>The -lines option displays source information, if available.</p> <p>You can use this command to format and view the <code>_stackTrace</code> field, which is a binary array.</p>
ProcInfo [-env] [-time] [-mem]	<p>Displays environment variables for the process, kernel CPU time, and memory usage statistics.</p>
RCWCleanupList <RCWCleanupList address>	<p>Displays the list of runtime callable wrappers at the specified address that are awaiting cleanup.</p>
SaveModule <Base address> <Filename>	<p>Writes an image, which is loaded in memory at the specified address, to the specified file.</p>
SOSFlush	<p>Flushes an internal SOS cache.</p>
StopOnException [-derived] [-create -create2] <Exception> <Pseudo-register number>	<p>Causes the debugger to stop when the specified exception is thrown, but to continue running when other exceptions are thrown.</p> <p>The -derived option catches the specified exception and every exception that derives from the specified exception.</p>
SyncBlk [-all <syncblk number>]	<p>Displays the specified <code>SyncBlock</code> structure or all <code>SyncBlock</code> structures. If you do not pass any arguments, the SyncBlk command displays the <code>SyncBlock</code> structure corresponding to objects that are owned by a thread.</p>

Command	Description
	A <code>SyncBlock</code> structure is a container for extra information that does not need to be created for every object. It can hold COM interop data, hash codes, and locking information for thread-safe operations.
ThreadPool	Displays information about the managed thread pool, including the number of work requests in the queue, the number of completion port threads, and the number of timers.
Token2EE < <i>module name</i> > < <i>token</i> >	<p>Turns the specified metadata token in the specified module into a <code>MethodTable</code> structure or <code>MethodDesc</code> structure.</p> <p>You can pass <code>*</code> for the module name parameter to find what that token maps to in every loaded managed module. You can also pass the debugger's name for a module, such as <code>mscorlib</code> or <code>image00400000</code>.</p>
Threads [-live] [-special]	<p>Displays all managed threads in the process.</p> <p>The Threads command displays the debugger shorthand ID, the CLR thread ID, and the operating system thread ID. Additionally, the Threads command displays a Domain column that indicates the application domain in which a thread is executing, an APT column that displays the COM apartment mode, and an Exception column that displays the last exception thrown in the thread.</p> <p>The -live option displays threads associated with a live thread.</p> <p>The -special option displays all special threads created by the CLR. Special threads include garbage collection threads (in concurrent and server garbage collection), debugger helper threads, finalizer threads, AppDomain unload threads, and thread pool timer threads.</p>
ThreadState < <i>State value field</i> >	<p>Displays the state of the thread. The <code>value</code> parameter is the value of the <code>State</code> field in the Threads report output.</p> <p>Example:</p> <pre>0:003> !Threads ThreadCount: 2 UnstartedThread: 0 BackgroundThread: 1 PendingThread: 0 DeadThread: 0 Hosted Runtime: no PreEmptive GC Alloc Lock ID OSID ThreadOBJ State GC Context Domain Count APT Exception 0 1 250 0019b068 a020 Disabled 02349668:02349fe8 0015def0 0 MTA 2 2 944 001a6020 b220 Enabled 00000000:00000000 0015def0 0 MTA (Finalizer) 0:003> !ThreadState b220 Legal to Join Background CLR Owns CoInitialized In Multi Threaded Apartment</pre>
TraverseHeap [-xml] < <i>filename</i> >	Writes heap information to the specified file in a format understood by the CLR profiler. The -xml option causes the TraverseHeap command to format the file as XML.

Command	Description
U [-gcinfo] [-ehinfo] [-n] <MethodDesc address> <Code address>	<p>Displays an annotated disassembly of a managed method specified either by a <code>MethodDesc</code> structure pointer for the method or by a code address within the method body. The U command displays the entire method from start to finish, with annotations that convert metadata tokens to names.</p> <p>The -gcinfo option causes the U command to display the <code>GCInfo</code> structure for the method.</p> <p>The -ehinfo option displays exception information for the method. You can also obtain this information with the EHInfo command.</p> <p>The -n option disables the display of source file names and line numbers. If the debugger has the option SYMOPT_LOAD_LINES specified, SOS looks up the symbols for every managed frame and, if successful, displays the corresponding source file name and line number. You can specify the -n option to disable this behavior.</p>
VerifyHeap	<p>Checks the garbage collector heap for signs of corruption and displays any errors found.</p> <p>Heap corruptions can be caused by platform invoke calls that are constructed incorrectly.</p>
VerifyObj <object address>	<p>Checks the object that is passed as an argument for signs of corruption.</p>
VMMMap	<p>Traverses the virtual address space and displays the type of protection applied to each region.</p>
VMStat	<p>Provides a summary view of the virtual address space, ordered by each type of protection applied to that memory (free, reserved, committed, private, mapped, image). The TOTAL column displays the result of the AVERAGE column multiplied by the BLK COUNT column.</p>

Remarks

The SOS Debugging Extension lets you view information about code that is running inside the CLR. For example, you can use the SOS Debugging Extension to display information about the managed heap, look for heap corruptions, display internal data types used by the runtime, and view information about all managed code running inside the runtime.

To use the SOS Debugging Extension in Visual Studio, install the [Windows Driver Kit \(WDK\)](#). For information about the integrated debugging environment in Visual Studio,

see [Debugging Environments](#).

You can also use the SOS Debugging Extension by loading it into the [WinDbg.exe debugger](#) and executing commands within WinDbg.exe.

To load the SOS Debugging Extension into the WinDbg.exe debugger, run the following command in the tool:

```
Console  
.loadby sos clr
```

WinDbg.exe and Visual Studio use a version of SOS.dll that corresponds to the version of Mscorwks.dll currently in use. By default, you should use the version of SOS.dll that matches the current version of Mscorwks.dll.

To use a dump file created on another computer, make sure that the Mscorwks.dll file that came with that installation is in your symbol path, and load the corresponding version of SOS.dll.

To load a specific version of SOS.dll, type the following command into the Windows Debugger:

```
Console  
.load <full path to sos.dll>
```

Examples

The following command displays the contents of an array at the address `00ad28d0`. The display starts from the second element and continues for five elements.

```
Console  
!dumparray -start 2 -length 5 -detail 00ad28d0
```

The following command displays the contents of an assembly at the address `1ca248`.

```
Console  
!dumpassembly 1ca248
```

The following command displays information about the garbage collector heap.

```
Console
```

```
!dumpheap
```

The following command writes the contents of the in-memory stress log to a (default) file called StressLog.txt in the current directory.

```
Console
```

```
!DumpLog
```

The following command displays the `MethodDesc` structure at the address `902f40`.

```
Console
```

```
!dumpmd 902f40
```

The following command displays information about a module at the address `1caa50`.

```
Console
```

```
!dumpmodule 1caa50
```

The following command displays information about an object at the address `a79d40`.

```
Console
```

```
!DumpObj a79d40
```

The following command displays the fields of a value class at the address `00a79d9c` using the method table at the address `0090320c`.

```
Console
```

```
!DumpVC 0090320c 00a79d9c
```

The following command displays the process memory used by the garbage collector.

```
Console
```

```
!eeheap -gc
```

The following command displays all objects scheduled for finalization.

```
Console
```

```
!finalizequeue
```

The following command determines the application domain of an object at the address `00a79d98`.

```
Console
```

```
!findappdomain 00a79d98
```

The following command displays all garbage collector handles in the current process.

```
Console
```

```
!gcinfo 5b68dbb8
```

The following command displays the `MethodTable` and `EEClass` structures for the `Main` method in the class `MainClass` in the module `unittest.exe`.

```
Console
```

```
!name2ee unittest.exe MainClass.Main
```

The following command displays information about the metadata token at the address `02000003` in the module `unittest.exe`.

```
Console
```

```
!token2ee unittest.exe 02000003
```

See also

- [Tools](#)
- [Developer command-line shells](#)



Collaborate with us on

.NET

.NET feedback

GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlMetal.exe (Code Generation Tool)

Article • 02/13/2023

The SqlMetal command-line tool generates code and mapping for the LINQ to SQL component of the .NET Framework. By applying options that appear later in this topic, you can instruct SqlMetal to perform several different actions that include the following:

- From a database, generate source code and mapping attributes or a mapping file.
- From a database, generate an intermediate database markup language (.dbml) file for customization.
- From a .dbml file, generate code and mapping attributes or a mapping file.

This tool is automatically installed with Visual Studio 2019 and earlier versions. By default, the file is located at `%ProgramFiles%\Microsoft SDKs\Windows\[version]\bin`. If you don't install Visual Studio, you can also get the SQLMetal file by downloading the [Windows SDK](#).

ⓘ Note

Developers who use Visual Studio can also use the Object Relational Designer to generate entity classes. The command-line approach scales well for large databases. Because SqlMetal is a command-line tool, you can use it in a build process.

To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#). At the command prompt, enter the following command:

Console

```
sqlmetal [options] [<input file>]
```

Options

To view the most current option list, type `sqlmetal /?` at a command prompt from the installed location.

Connection Options

Option	Description
--------	-------------

Option	Description
/server: <i><name></i>	Specifies database server name.
/database: <i><name></i>	Specifies database catalog on server.
/user: <i><name></i>	Specifies logon user id. Default value: Use Windows authentication.
/password: <i><password></i>	Specifies logon password. Default value: Use Windows authentication.
/conn: <i><connection string></i>	Specifies database connection string. Cannot be used with /server, /database, /user, or /password options. Do not include the file name in the connection string. Instead, add the file name to the command line as the input file. For example, the following line specifies "c:\northwnd.mdf" as the input file: sqlmetal /code:"c:\northwind.cs" /language:csharp "c:\northwnd.mdf" .
/timeout: <i><seconds></i>	Specifies time-out value when SqlMetal accesses the database. Default value: 0 (that is, no time limit).

Extraction options

Option	Description
/views	Extracts database views.
/functions	Extracts database functions.
/sprocs	Extracts stored procedures.

Output options

Option	Description
/dbml [:file]	Sends output as .dbml. Cannot be used with /map option.
/code [:file]	Sends output as source code. Cannot be used with /dbml option.
/map [:file]	Generates an XML mapping file instead of attributes. Cannot be used with /dbml option.

Miscellaneous

Option	Description
/language: <language>	Specifies source code language. Valid <language>: vb, csharp. Default value: Derived from extension on code file name.
/namespace: <name>	Specifies namespace of the generated code. Default value: no namespace.
/context: <type>	Specifies name of data context class. Default value: Derived from database name.
/entitybase: <type>	Specifies the base class of the entity classes in the generated code. Default value: Entities have no base class.
/pluralize	Automatically pluralizes or singularizes class and member names. This option is available only in the U.S. English version.
/serialization: <option>	Generates serializable classes. Valid <option>: None, Unidirectional. Default value: None. For more information, see Serialization .

Input File

Option	Description
<input file>	Specifies a SQL Server Express .mdf file, a SQL Server Compact 3.5 .sdf file, or a .dbml intermediate file.

Remarks

SqlMetal functionality actually involves two steps:

- Extracting the metadata of the database into a .dbml file.
- Generating a code output file.

By using the appropriate command-line options, you can produce Visual Basic or C# source code, or you can produce an XML mapping file.

To extract the metadata from an .mdf file, you must specify the name of the .mdf file after all other options.

If no **/server** is specified, **localhost/sqlexpress** is assumed.

Microsoft SQL Server 2005 throws an exception if one or more of the following conditions are true:

- SqlMetal tries to extract a stored procedure that calls itself.
- The nesting level of a stored procedure, function, or view exceeds 32.

SqlMetal catches this exception and reports it as a warning.

To specify an input file name, add the name to the command line as the input file. Including the file name in the connection string (using the **/conn** option) is not supported.

Examples

Generate a .dbml file that includes extracted SQL metadata:

```
sqlmetal /server:myserver /database:northwind /dbml:mymeta.dbml
```

Generate a .dbml file that includes extracted SQL metadata from an .mdf file by using SQL Server Express:

```
sqlmetal /dbml:mymeta.dbml mydbfile.mdf
```

Generate a .dbml file that includes extracted SQL metadata from SQL Server Express:

```
sqlmetal /server:.\sqlexpress /dbml:mymeta.dbml /database:northwind
```

Generate source code from a .dbml metadata file:

```
sqlmetal /namespace:nwind /code:nwind.cs /language:csharp mymeta.dbml
```

Generate source code from SQL metadata directly:

```
sqlmetal /server:myserver /database:northwind /namespace:nwind /code:nwind.cs  
/language:csharp
```

Note

When you use the **/pluralize** option with the Northwind sample database, note the following behavior. When SqlMetal makes row-type names for tables, the table names are singular. When it makes **DataContext** properties for tables, the table names are plural. Coincidentally, the tables in the Northwind sample database are

already plural. Therefore, you do not see that part working. Although it is common practice to name database tables singular, it is also a common practice in .NET to name collections plural.

See also

- [How to: Generate the Object Model in Visual Basic or C#](#)
- [Code Generation in LINQ to SQL](#)
- [External Mapping](#)

Storeadm.exe (Isolated Storage Tool)

Article • 07/23/2022

The Isolated Storage tool lists or removes all existing stores for the current user.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
storeadm [/list][/machine][/remove][/roaming][/quiet]
```

Parameters

Option	Description
/h[elp]	Displays command syntax and options for the tool.
/list	Displays all existing stores for the current user. This includes the stores for all applications or assemblies executed by this user.
/machine	Selects the machine store. Use this option with the /list or /remove option to specify that the action should apply to the machine store. New in .NET Framework 2.0
/quiet	Specifies quiet mode; suppresses informational output so that only error messages appear.
/remove	Permanently removes all existing stores for the current user.
/roaming	Selects the roaming store. Use this option with the /list or /remove options to specify that the action should apply to the roaming store.
/?	Displays command syntax and options for the tool.

Remarks

Running Storeadm.exe from the command line without specifying any options displays the syntax and options for the tool.

The **/list** and **/remove** options are typically used one at a time; however, if two or more options are specified they will be performed in the order in which they appear on the command line.

Applications have a choice of saving to one of two stores for a user or to the machine store:

- The local store exists in a location that is guaranteed not to roam, even if user data roaming is enabled for the user.
- The roaming store exists in a location that is able to roam, but can only do so if roaming is enabled for the user via Windows administration.
- The machine store is common to all users on a machine and is stored under a common directory on that machine.

Whether roaming is actually enabled for the user does not affect the administration of Storeadm.exe. Running the tool without any options applies all actions to the local store. Running the tool with the **/roaming** option applies all actions to the store that is able to roam. Running the tool with the **/machine** option applies all actions to the machine store.

See also

- [Tools](#)
- [Isolated Storage](#)
- [Developer command-line shells](#)

Tlbexp.exe (Type Library Exporter)

Article • 07/23/2022

The Type Library Exporter generates a type library that describes the types defined in a common language runtime assembly.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
tlbexp assemblyName [options]
```

Parameters

[+] Expand table

Argument	Description
<i>assemblyName</i>	The assembly for which to export a type library.

[+] Expand table

Option	Description
<i>/asmpath:directory</i>	Specifies the location to search for assemblies. If you use this option, you must explicitly specify the locations to search for referenced assemblies, including the current directory. When you use the asmpath option, the Type Library Exporter will not look for an assembly in the global assembly cache (GAC).
<i>/help</i>	Displays command syntax and options for the tool.
<i>/names:filename</i>	Specifies the capitalization of names in a type library. The <i>filename</i> argument is a text file. Each line in the file specifies the capitalization of one name in the type library.
<i>/nologo</i>	Suppresses the Microsoft startup banner display.

Option	Description
/oldnames	Forces Tlbexp.exe to export decorated type names if there is a type name conflict. Note that this was the default behavior in versions prior to .NET Framework version 2.0.
/out: file	Specifies the name of the type library file to generate. If you omit this option, Tlbexp.exe generates a type library with the same name as the assembly (the actual assembly name, which might not necessarily be the same as the file containing the assembly) and a .tlb extension.
/silence: warningnumber	Suppresses the display of the specified warning. This option cannot be used with /silent .
/silent	Suppresses the display of success messages. This option cannot be used with /silence .
/tlbreference: typelibraryname	Forces Tlbexp.exe to explicitly resolve type library references without consulting the registry. For example, if assembly B references assembly A, you can use this option to provide an explicit type library reference, rather than relying on the type library specified in the registry. Tlbexp.exe performs a version check to ensure that the type library version matches the assembly version; otherwise, it generates an error.
	Note that the tlbreference option still consults the registry in cases where the ComImportAttribute attribute is applied to an interface that is then implemented by another type.
/tlbrefpath: path	Fully qualified path to a referenced type library.
/win32	When compiling on a 64-bit computer, this option specifies that Tlbexp.exe generates a 32-bit type library.
/win64	When compiling on a 32-bit computer, this option specifies that Tlbexp.exe generates a 64-bit type library.
/verbose	Specifies verbose mode; displays a list of any referenced assemblies for which a type library needs to be generated.
/?	Displays command syntax and options for the tool.

ⓘ Note

The command-line options for Tlbexp.exe are case-insensitive and can be supplied in any order. You only need to specify enough of the option to uniquely identify it. For example, **/n** is equivalent to **/nologo**, and **/o: outfile.tlb** is equivalent to **/out: outfile.tlb**.

Remarks

Tlbexp.exe generates a type library that contains definitions of the types defined in the assembly. Applications such as Visual Basic 6.0 can use the generated type library to bind to the .NET types defined in the assembly.

ⓘ Important

You cannot use Tlbexp.exe to export Windows metadata (.winmd) files. Exporting Windows Runtime assemblies is not supported.

The entire assembly is converted at once. You cannot use Tlbexp.exe to generate type information for a subset of the types defined in an assembly.

You cannot use Tlbexp.exe to produce a type library from an assembly that was imported using the [Type Library Importer \(Tlbimp.exe\)](#). Instead, you should refer to the original type library that was imported with Tlbimp.exe. You can export a type library from an assembly that references assemblies that were imported using Tlbimp.exe. See the examples section below.

Tlbexp.exe places generated type libraries in the current working directory or the directory specified for the output file. A single assembly might cause several type libraries to be generated.

Tlbexp.exe generates a type library but does not register it. This is in contrast to the [Assembly Registration tool \(Regasm.exe\)](#), which both generates and registers a type library. To generate and register a type library with COM, use Regasm.exe.

If you do not specify either the `/win32` or `/win64` option, Tlbexp.exe generates a 32-bit or 64-bit type library that corresponds to the type of computer on which you are performing the compilation (32-bit or 64-bit computer). For cross-compilation purposes, you can use the `/win64` option on a 32-bit computer to generate a 64-bit type library and you can use the `/win32` option on a 64-bit computer to generate a 32-bit type library. In 32-bit type libraries, the `SYSKIND` value is set to `SYS_WIN32`. In 64-bit type libraries, the `SYSKIND` value is set to `SYS_WIN64`. All data type transformations (for example, pointer-sized data types such as `IntPtr` and `UIntPtr`) are converted appropriately.

If you use the `MarshalAsAttribute` attribute to specify a `SafeArraySubType` value of `VT_UNKNOWN` or `VT_DISPATCH`, Tlbexp.exe ignores any subsequent use of the `SafeArrayUserDefinedSubType` field. For example, given the following signatures:

C#

```
[return:MarshalAs(UnmanagedType.SafeArray,
SafeArraySubType=VarEnum.VT_UNKNOWN,
SafeArrayUserDefinedSubType=typeof(ConsoleKeyInfo))] public Array
StructUnkSafe(){return null;}
[return:MarshalAs(UnmanagedType.SafeArray,
SafeArraySubType=VarEnum.VT_DISPATCH,
SafeArrayUserDefinedSubType=typeof(ConsoleKeyInfo))] public Array
StructDispSafe(){return null;}
```

the following type library is generated:

C++

```
[id(0x60020004)]
HRESULT StructUnkSafe([out, retval] SAFEARRAY(IUnknown**)* pRetVal);
[id(0x60020005)]
HRESULT StructDispSafe([out, retval] SAFEARRAY(IDispatch**)* pRetVal);
```

Note that Tlbexp.exe ignores the `SafeArrayUserDefinedSubType` field.

Because type libraries cannot accommodate all the information found in assemblies, Tlbexp.exe might discard some data during the export process. For an explanation of the transformation process and identification of the source of each piece of information emitted to a type library, see the [Assembly to Type Library Conversion Summary](#).

Note that the Type Library Exporter exports methods that have `TypedReference` parameters as `VARIANT`, even though the `TypedReference` object has no meaning in unmanaged code. When you export methods that have `TypedReference` parameters, the Type Library Exporter will not generate a warning or error and unmanaged code that uses the resulting type library will not run properly.

Examples

The following command generates a type library with the same name as the assembly found in `myTest.dll`.

Console

```
tlbexp myTest.dll
```

The following command generates a type library with the name `clipper.tlb`.

Console

```
tlbexp myTest.dll /out:clipper.tlb
```

The following example illustrates using Tlbexp.exe to export a type library from an assembly that references assemblies that were imported using Tlbimp.exe.

First use Tlbimp.exe to import the type library `myLib.tlb` and save it as `myLib.dll`.

Console

```
tlbimp myLib.tlb /out:myLib.dll
```

The following command uses the C# compiler to compile the `Sample.cs`, which references `myLib.dll` created in the previous example.

Console

```
CSC Sample.cs /reference:myLib.dll /out:Sample.dll
```

The following command generates a type library for `Sample.dll` that references `myLib.dll`.

Console

```
tlbexp Sample.dll
```

See also

- [TypeLibExporterFlags](#)
- [Tools](#)
- [Regasm.exe \(Assembly Registration Tool\)](#)
- [Assembly to Type Library Conversion Summary](#)
- [Tlbimp.exe \(Type Library Importer\)](#)
- [Developer command-line shells](#)



Collaborate with us on
GitHub

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

Tlbimp.exe (Type Library Importer)

Article • 09/15/2021

The Type Library Importer converts the type definitions found within a COM type library into equivalent definitions in a common language runtime assembly. The output of Tlbimp.exe is a binary file (an assembly) that contains runtime metadata for the types defined within the original type library. You can examine this file with tools such as [Ildasm.exe](#).

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
tlbimp tlbFile [options]
```

Parameters

Argument	Description
<code>tlbFile</code>	The name of any file that contains a COM type library.

Option	Description
<code>/asmversion: versionnumber</code>	Specifies the version number of the assembly to produce. Specify <i>versionnumber</i> in the format <i>major.minor.build.revision</i> .
<code>/company: companyinformation</code>	Adds company information to the output assembly.
<code>/copyright: copyrightinformation</code>	Adds copyright information to the output assembly. This information can be viewed in the File Properties dialog box for the assembly.
<code>/delsign</code>	Specifies to Tlbimp.exe to sign the resulting assembly with a strong name using delayed signing. You must specify this option with either the <code>/keycontainer:</code> , <code>/keyfile:</code> , or <code>/publickey:</code> option. For more information on the delayed signing process, see Delay Signing an Assembly .

Option	Description
<code>/help</code>	Displays command syntax and options for the tool.
<code>/keycontainer:</code> <i>containername</i>	Signs the resulting assembly with a strong name using the public/private key pair found in the key container specified by <i>containername</i> .
<code>/keyfile:</code> <i>filename</i>	Signs the resulting assembly with a strong name using the publisher's official public/private key pair found in <i>filename</i> .
<code>/machine:</code> <i>machinetype</i>	Creates an assembly that targets the specified machine type (microprocessor). Supported machine types: x86, x64, Itanium, and Agnostic.
<code>/namespace:</code> <i>namespace</i>	Specifies the namespace in which to produce the assembly.
<code>/noclassmembers</code>	Prevents Tlbimp.exe from adding members to classes. This avoids a potential TypeLoadException .
<code>/nologo</code>	Suppresses the Microsoft startup banner display.
<code>/out:</code> <i>filename</i>	Specifies the name of the output file, assembly, and namespace in which to write the metadata definitions. The <code>/out</code> option has no effect on the assembly's namespace if the type library specifies the Interface Definition Language (IDL) custom attribute that explicitly controls the assembly's namespace. If you do not specify this option, Tlbimp.exe writes the metadata to a file with the same name as the actual type library defined within the input file and assigns it a .dll extension. If the output file is the same name as the input file, the tool generates an error to prevent overwriting the type library.
<code>/primary</code>	Produces a primary interop assembly for the specified type library. Information is added to the assembly indicating that the publisher of the type library produced the assembly. By specifying a primary interop assembly, you differentiate a publisher's assembly from any other assemblies that are created from the type library using Tlbimp.exe. You should only use the <code>/primary</code> option if you are the publisher of the type library that you are importing with Tlbimp.exe. Note that you must sign a primary interop assembly with a strong name . For more information, see Primary Interop Assemblies .
<code>/product:</code> <i>productinformation</i>	Adds product information to the output assembly. This information can be viewed in the File Properties dialog box for the assembly.
<code>/productversion:</code> <i>productversioninformation</i>	Adds product version information to the output assembly. There are no format restrictions. This information can be viewed in the File Properties dialog box for the assembly.

Option	Description
/publickey: <i>filename</i>	Specifies the file containing the public key to use to sign the resulting assembly. If you specify the /keyfile: or /keycontainer: option instead of /publickey: , Tlbimp.exe generates the public key from the public/private key pair supplied with /keyfile: or /keycontainer: . The /publickey: option supports test key and delay signing scenarios. The file is in the format generated by Sn.exe. For more information, see the -p option of Sn.exe in Strong Name Tool (Sn.exe) .
/reference: <i>filename</i>	Specifies the assembly file to use to resolve references to types defined outside the current type library. If you do not specify the /reference option, Tlbimp.exe automatically recursively imports any external type library that the type library being imported references. If you specify the /reference option, the tool attempts to resolve external types in the referenced assemblies before it imports other type libraries.
/silence: <i>warningnumber</i>	Suppresses the display of the specified warning. This option cannot be used with /silent .
/silent	Suppresses the display of success messages. This option cannot be used with /silence .
/strictref	Does not import a type library if the tool cannot resolve all references within the current assembly, the assemblies specified with the /reference option, or registered primary interop assemblies (PIAs).
/strictref:nopia	Same as /strictref , but ignores PIAs.
/sysarray	Specifies to the tool to import a COM style SafeArray as a managed Array type.
/tlbreference: <i>filename</i>	<p>Specifies the type library file to use to resolve type library references without consulting the registry.</p> <p>Note that this option will not load some older type library formats. However, you can still load older type library formats implicitly through the registry or current directory.</p>
/trademark: <i>trademarkinformation</i>	Adds trademark information to the output assembly. This information can be viewed in the File Properties dialog box for the assembly.

Option	Description
<code>/transform: transformname</code>	<p>Transforms metadata as specified by the <i>transformname</i> parameter.</p> <p>Specify dispret for the <i>transformname</i> parameter to transform [out, retval] parameters of methods on dispatch-only interfaces (dispinterfaces) into return values.</p> <p>For more information about this option, see the examples later in this topic.</p>
<code>/unsafe</code>	<p>Produces interfaces without .NET Framework security checks. Calling a method that is exposed in this way might pose a security risk. You should not use this option unless you are aware of the risks of exposing such code.</p>
<code>/verbose</code>	<p>Specifies verbose mode; displays additional information about the imported type library.</p>
<code>/VariantBoolFieldToBool</code>	<p>Converts <code>VARIANT_BOOL</code> fields in structures to <code>Boolean</code>.</p>
<code>/?</code>	<p>Displays command syntax and options for the tool.</p>

ⓘ Note

The command-line options for Tlbimp.exe are case-insensitive and can be supplied in any order. You only need to specify enough of the option to uniquely identify it. Therefore, `/n` is equivalent to `/nologo` and `/ou: outfile.dll` is equivalent to `/out: outfile.dll`.

Remarks

Tlbimp.exe performs conversions on an entire type library at one time. You cannot use the tool to generate type information for a subset of the types defined within a single type library.

It is often useful or necessary to be able to assign [strong names](#) to assemblies. Therefore, Tlbimp.exe includes options for supplying the information necessary to generate strongly named assemblies. Both the `/keyfile:` and `/keycontainer:` options sign assemblies with strong names. Therefore, it is logical to supply only one of these options at a time.

You can specify multiple reference assemblies by using the `/reference` option multiple times.

Due to the way in which Tlbimp.exe generates assemblies, it is not possible to retarget an assembly to a different `mscorlib` version. For example, if you desire to generate an assembly that targets .NET Framework 2.0, the Tlbimp.exe shipped with the .NET Framework 2.0/3.0/3.5 SDK must be used. In order to target .NET Framework 4.x, the Tlbimp.exe shipped with a .NET Framework 4.x SDK should be used.

A resource ID can optionally be appended to a type library file when importing a type library from a module containing multiple type libraries. Tlbimp.exe is able to locate this file only if it is in the current directory or if you specify the full path. See the example later in this topic.

Examples

The following command generates an assembly with the same name as the type library found in `myTest.tlb` and with the `.dll` extension.

```
Console  
tlbimp myTest.tlb
```

The following command generates an assembly with the name `myTest.dll`.

```
Console  
tlbimp myTest.tlb /out:myTest.dll
```

The following command generates an assembly with the same name as the type library specified by `MyModule.dll\1` and with the `.dll` extension. `MyModule.dll\1` must be located in the current directory.

```
Console  
tlbimp MyModule.dll\1
```

The following command generates an assembly with the name `myTestLib.dll` for the type library `TestLib.dll`. The `/transform:dispret` option transforms any [out, retval] parameters of methods on dispinterfaces in the type library into return values in the managed library.

```
Console  
tlbimp TestLib.dll /transform:dispret /out:myTestLib.dll
```

The type library `TestLib.dll`, in the preceding example, includes a dispinterface method named `SomeMethod` that returns void and has an [out, retval] parameter. The following code is the input type library method signature for `SomeMethod` in `TestLib.dll`.

C++

```
void SomeMethod([out, retval] VARIANT_BOOL*);
```

Specifying the `/transform:dispret` option causes `Tlbimp.exe` to transform the `[out, retval]` parameter of `SomeMethod` into a `bool` return value. The following is the method signature that `Tlbimp.exe` produces for `SomeMethod` in the managed library `myTestLib.dll` when the `/transform:dispret` option is specified.

C#

```
bool SomeMethod();
```

If you use `Tlbimp.exe` to produce a managed library for `TestLib.dll` without specifying the `/transform:dispret`, the tool produces the following method signature for `SomeMethod` in the managed library `myTestLib.dll`.

C#

```
void SomeMethod(out bool x);
```

See also

- [Tools](#)
- [Tlbexp.exe \(Type Library Exporter\)](#)
- [Importing a Type Library as an Assembly](#)
- [Type Library to Assembly Conversion Summary](#)
- [Ildasm.exe \(IL Disassembler\)](#)
- [Sn.exe \(Strong Name Tool\)](#)
- [Strong-Named Assemblies](#)
- [Attributes for Importing Type Libraries into Interop Assemblies](#)
- [Developer command-line shells](#)

Winmdexp.exe (Windows Runtime Metadata Export Tool)

Article • 07/23/2022

The Windows Runtime Metadata Export Tool (Winmdexp.exe) transforms a .NET Framework module into a file that contains Windows Runtime metadata. Although .NET Framework assemblies and Windows Runtime metadata files use the same physical format, there are differences in the content of the metadata tables, which means that .NET Framework assemblies are not automatically usable as Windows Runtime Components. The process of turning a .NET Framework module into a Windows Runtime component is referred to as *exporting*. In .NET Framework 4.5 and 4.5.1, the resulting Windows metadata (.winmd) file contains both metadata and implementation.

When you use the **Windows Runtime Component** template, which is located under **Windows Store** for C# and Visual Basic in Visual Studio 2013 or Visual Studio 2012, the compiler target is a .winmdobj file, and a subsequent build step calls Winmdexp.exe to export the .winmdobj file to a .winmd file. This is the recommended way to build a Windows Runtime component. Use Winmdexp.exe directly when you want more control over the build process than Visual Studio provides.

This tool is automatically installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

At the command prompt, type the following:

Syntax

Console

```
winmdexp [options] winmdmodule
```

Parameters

Argument or option	Description
<code>winmdmodule</code>	Specifies the module (.winmdobj) to be exported. Only one module is allowed. To create this module, use the <code>/target</code> compiler option with the <code>winmdobj</code> target. See -target:winmdobj (C# Compiler Options) or -target (Visual Basic) .

Argument or option	Description
/docfile: docfile /d: docfile	Specifies the output XML documentation file that Winmdexp.exe will produce. In .NET Framework 4.5, the output file is essentially the same as the input XML documentation file.
/moduledoc: docfile /md: docfile	Specifies the name of the XML documentation file that the compiler produced with <code>winmdmodule</code> .
/modulepdb: symbolfile /mp: symbolfile	Specifies the name of the program database (PDB) file that contains symbols for <code>winmdmodule</code> .
/nowarn: warning /out: file	Suppresses the specified warning number. For <code>warning</code> , supply only the numeric portion of the error code, without leading zeros. Specifies the name of the output Windows metadata (.winmd) file.
/pdb: symbolfile /p: symbolfile	Specifies the name of the output program database (PDB) file that will contain the symbols for the exported Windows metadata (.winmd) file.
/reference: winmd /r: winmd	Specifies a metadata file (.winmd or assembly) to reference during export. If you use the reference assemblies in "\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETCore\v4.5" ("\Program Files\..." on 32-bit computers), include references to both System.Runtime.dll and mscorlib.dll.
/utf8output	Specifies that output messages should be in UTF-8 encoding.
/warnaserror+	Specifies that all warnings should be treated as errors.
@ responsefile	Specifies a response (.rsp) file that contains options (and optionally <code>winmdmodule</code>). Each line in <code>responsefile</code> should contain a single argument or option.

Remarks

Winmdexp.exe is not designed to convert an arbitrary .NET Framework assembly to a .winmd file. It requires a module that is compiled with the `/target:winmdobj` option, and

additional restrictions apply. The most important of these restrictions is that all types that are exposed in the API surface of the assembly must be Windows Runtime types. For more information, see the "Declaring types in Windows Runtime Components" section of the article [Creating Windows Runtime Components in C# and Visual Basic](#).

When you write a Windows 8.x Store app or a Windows Runtime component with C# or Visual Basic, the .NET Framework provides support to make programming with the Windows Runtime more natural. This is discussed in the article [.NET Framework Support for Windows Store Apps and Windows Runtime](#). In the process, some commonly used Windows Runtime types are mapped to .NET Framework types. Winmdexp.exe reverses this process and produces an API surface that uses the corresponding Windows Runtime types. For example, types that are constructed from the `IList<T>` interface map to types that are constructed from the Windows Runtime `IVector<T>` interface.

See also

- [.NET Framework Support for Windows Store Apps and Windows Runtime](#)
- [Creating Windows Runtime Components in C# and Visual Basic](#)
- [Winmdexp.exe Error Messages](#)
- [Build, Deployment, and Configuration Tools \(.NET Framework\)](#)

Winmdexp.exe error messages

Article • 09/15/2021

The build process calls [Winmdexp.exe \(Windows Runtime Metadata Export Tool\)](#) when you use the **Windows Runtime Component** template in Visual Studio 2012, so Winmdexp.exe error messages appear in the **Error List**. Winmdexp.exe operates on a module that is compiled with the `/target:winmdobj` option. Because it requires a compiled module as input, its error messages don't appear unless compilation succeeds.

The error messages are designed to contain all the information you need to address the error conditions they report. However, some problems require more information than will fit in the message. You can find additional information in [Diagnosing Windows Runtime component error conditions](#).

If your error is not discussed in that article, and you feel that the message doesn't contain sufficient information to address the issue, use the feedback link in that article and include the error message. Alternatively, you can file a bug at the [Developer Community website](#) . You can also look for more information on the [Microsoft Forums](#) .

See also

- [Winmdexp.exe \(Windows Runtime Metadata Export Tool\)](#)
- [Diagnosing Windows Runtime component error conditions](#)

Winres.exe (Windows Resource Localization Editor)

Article • 05/04/2023

The Windows Resource Localization Editor, Winres.exe, is a visual layout tool that helps localization experts localize Windows Forms user interface (UI) resources used by forms. You can create the *.resx* or *.resources* files that are used as input to Winres.exe using a visual design environment such as Microsoft Visual Studio. For information on deploying resources in .NET Framework applications, see [Resources in .NET apps](#).

Winres.exe is installed with Visual Studio. To run the tool, use [Visual Studio Developer Command Prompt](#) or [Visual Studio Developer PowerShell](#).

⚠ Warning

Do not open untrusted files with this tool. If the file is a binary file, it will be deserialized using binary deserialization, which is unsafe. For more information, see [Deserialization risks in use of BinaryFormatter and related types](#).

Syntax

Console

```
winres resourceFile  
winres /?
```

Arguments

Argument	Description
<code>resourceFile</code>	The resource file to localize. This file must be a Windows Forms form <i>.resx</i> or <i>.resources</i> file generated by the Visual Studio designer. Winres.exe cannot open generic <i>.resx</i> or <i>.resources</i> files.

Option	Description
<code>/?</code>	Displays command syntax and options for the tool.

Remarks

The state of UI elements from a form in a Windows Forms project is typically stored in resource files, which are either XML-based files with the extension .resx or the corresponding compiled, binary versions with the extension .resources. Winres.exe is a tool that enables limited editing of either type of file outside of the Visual Studio design environment. Specifically, it allows the following types of editing operations:

- A neutral or specific culture resource file can be edited to change the UI properties of the form or its controls, such as their text, size, or position.
- Neutral or specific culture resource files can be generated from the default resource file.
- A culture resource file can be saved as another culture resource file. For example, an English (U.S.) resource file could be saved as a Polish resource file. Typically the new file would subsequently be edited to be compatible with the new culture.

Also see [Hierarchical Organization of Resources for Localization](#) or [Hierarchical Organization of Resources for Localization](#).

Winres.exe cannot convert a .resx file into its corresponding .resources file; use the Resgen.exe tool instead. For more information about Resgen.exe, see [Resgen.exe \(Resource File Generator\)](#).

Winres.exe is a graphical application that recreates a design-time version of a Windows Forms form from just the resource file, without having access to the source code.

Winres.exe hosts Visual Studio's **Windows Forms Form Designer** and **Properties** window. These features enable visual editing of a .resources or .resx file containing a Windows Forms form. Typically, localizers use Winres.exe to edit control labels and adjust the location and size of controls to accommodate the labels for the target culture.

If Winres.exe cannot resolve the type of a control, it creates a placeholder control in the localized .resx or .resources file. The placeholder control appears on the Windows Forms form as a hatched window. The size and position of the hatched window matches that of the actual control. All the available localizable properties for the placeholder control appear in the **Properties** window. Any changes that you make to the placeholder control are saved for the actual control.

Winres.exe versus Visual Studio

In general, before you begin to localize an application's Windows Forms forms, you should decide whether you want to use Visual Studio or Winres.exe as the localization

tool. Version compatibility, as described later, may prevent you from switching from one tool to the other.

The advantage of Visual Studio is that you can use it to both develop and localize an application. To localize a form, after development is complete, set the form's [LocalizableAttribute](#) (the **Localizable** property in the **Properties** editor) to `true` and change its **Language** property to the desired target culture. Then, edit strings and adjust the location and size of controls to accommodate the strings for the target culture. When you save the localized .resx file, Visual Studio writes only the localizable properties (properties that changed in the target culture) to the file. Visual Studio automatically creates a satellite assembly for the localized .resx file in the correct directory location.

Although Visual Studio provides an integrated development and localization environment, Winres.exe is the recommended tool to use if localization is done by third-party localizers. Because Winres.exe is a localization tool only, it allows for a cleaner separation of an application's code from the forms to be localized, which is more practical for managing large projects.

Using Winres.exe

To localize using Winres.exe, you must first develop an application using a visual designer like the **Windows Forms Designer** in Visual Studio. When development is complete, set the form's [LocalizableAttribute](#) (the **Localizable** property in the **Properties** editor) to `true`, and then hand off the .resx file for the default culture to a third-party localizer. This .resx file contains extra information that Winres.exe uses to recreate a design-time version of the original form.

Note

Winres.exe cannot be used to edit the default resource file. Winres.exe interprets all changed properties as localized properties and saves them to the target culture resource file.

The final versions of the culture resource files can finally be used to create localized versions of the application. For more information, see [Resources in .NET apps](#).

Winres.exe has the following features and capabilities:

- Winres can operate in Single File Mode (SFM) or Visual Studio File Mode (VSFM). SFM is the legacy mode where complete information about the form and its

contents is stored to the resource file. VSFM only stores only the cultural changes in the resource file.

- An error-reporting window, docked to the bottom-left of the main window.
- Hotkeys can be checked for duplicates: from the **Format** menu, click the **Check HotKeys** command.

Version compatibility

You should use the version of Winres.exe that was released with the .NET Framework you are using. The following table lists the compatible versions:

Visual Studio	.NET Framework	Winres.exe
Visual Studio .NET 2002	1.0	1.0
Visual Studio .NET 2003	1.1	1.1
Visual Studio 2005	2.0	2.0
Visual Studio 2008	3.0 and 3.5	3.0 and 3.5
Visual Studio 2010	4.0	4.0
Visual Studio 2017	4.6	4.6

ⓘ Note

Although VSFM has the advantage of being compatible with Visual Studio, since it stores only changed values in the resource file, Winres.exe requires that the parents of the current resource file be located in the same directory. For example, editing `TestApp.de-DE.resources`, a German in Germany resource file, requires the presence of the default resource file, `TestApp.resx`, and possibly the culture-neutral resource file, `TestApp.de.resources`.

Examples

To localize a .resx or .resources file associated with a form

1. To run Winres.exe, enter `winres` in the developer command prompt.

2. To open the default resources for a form to localize, select **File > Open** and navigate to the file to open it.

-or-

Specify the file to open at the command line when you start Winres.exe. The following command starts Winres.exe and loads the form associated with `TestApp.resx` in the Form Designer.

```
Console
winres TestApp.resx
```

The following command starts Winres.exe and loads the form associated with `TestApp.resources` in the Form Designer.

```
Console
winres TestApp.resources
```

① Note

If the form whose resources you are editing is an inherited form, both the assembly contained the inherited form and the assembly containing the inheriting (derived) form must either be registered in the Global Assembly Cache (GAC), or must reside in the same directory as WinRes.exe. For more information about installing .NET Framework components into the GAC, see [Global Assembly Cache](#).

3. Select controls on the form and change their **Text** and other properties to reflect the localized culture and its language. Move or resize controls as necessary to accommodate the localized text.
4. To save the localized version of the .resx or .resources file, click the **Save** icon or select **File > Save**. The tool displays the **Select Culture** window.
5. Select the appropriate culture and file mode then click **OK**.

The tool saves the file using the naming convention that the runtime expects for localized resource files. For example, if you localize `TestApp.resources` for German in Germany, the tool saves the file as `TestApp.de-DE.resources`. If you localize `TestApp.resx` for German in Germany, the tool saves the file as `TestApp.de-`

`DE.resx`. For more information about resource naming conventions, see [Package and Deploy resources](#). For a list of the predefined culture names used by the runtime, see the [CultureInfo](#) class.

See also

- [LocalizableAttribute](#)
- [CultureInfo](#)
- [ResourceManager](#)
- [ResourceReader](#)
- [ResourceWriter](#)
- [Tools](#)
- [Resources in .NET apps](#)
- [Globalization and localization](#)

Additional class libraries and APIs

Article • 06/12/2023

This article lists .NET Framework APIs that either were released out of band, target a specific platform, or are private or internal types.

OOB projects

To improve cross-platform development and introduce new functionality early, some .NET Framework features were released out of band (OOB).

[+] Expand table

Project	Description
System.Collections.Immutable	Provides collections that are thread safe and guaranteed to never change their contents.
WinHttpHandler	Provides a message handler for HttpClient based on the WinHTTP interface of Windows.
System.Numerics	Provides a library of vector types that can take advantage of SIMD hardware-based acceleration.
System.Threading.Tasks.Dataflow	The TPL Dataflow Library provides dataflow components to help increase the robustness of concurrency-enabled applications.

Platform-specific libraries

Some libraries target specific platforms. For example, the [CodePagesEncodingProvider](#) class makes code page encodings available to UWP apps developed using .NET Framework.

[+] Expand table

Project	Description
CodePagesEncodingProvider	Extends the EncodingProvider class to make code page encodings available to apps that target the Universal Windows Platform.
Point of Service for .NET	Provides a set of classes that enable you to develop applications that interact with POS devices.

Private APIs

These APIs support the product infrastructure and are not intended or supported to be used directly from your code.

- [Microsoft.SqlServer.Server.SmiOrderProperty.Item property](#)
- [System.Exception.PrepForRemoting method](#)
- [System.Data.SqlTypes.SqlBinary Constructor](#)
- [System.Data.SqlTypes.SqlChars.Stream property](#)
- [System.Data.SqlTypes.SqlGuid Constructor](#)
- [System.Data.SqlTypes.SqlMoney Constructor](#)
- [System.Data.SqlTypes.SqlMoney.ToSqlInternalRepresentation Method](#)
- [System.Data.SqlTypes.SqlStreamChars Constructor](#)
- [System.Data.SqlTypes.SqlStreamChars.CanSeek property](#)
- [System.Data.SqlTypes.SqlStreamChars.IsNull property](#)
- [System.Data.SqlTypes.SqlStreamChars.Length property](#)
- [System.Data.SqlTypes.SqlStreamChars.Close method](#)
- [System.Data.SqlTypes.SqlStreamChars.Dispose method](#)
- [System.Data.SqlTypes.SqlStreamChars.Flush method](#)
- [System.Data.SqlTypes.SqlStreamChars.Read method](#)
- [System.Data.SqlTypes.SqlStreamChars.Seek method](#)
- [System.Data.SqlTypes.SqlStreamChars.SetLength method](#)
- [System.Data.SqlTypes.SqlStreamChars.Write method](#)
- [System.IO.MemoryStream.InternalGetOriginAndLength method](#)
- [System.Net.ComNetOS class](#)
- [System.Net.Connection class](#)
- [System.Net.Connection.m_WriteList field](#)
- [System.Net.ConnectionGroup class](#)
- [System.Net.ConnectionGroup.m_ConnectionList field](#)
- [System.Net.ConnectStream.Connection property](#)
- [System.Net.CoreResponseData class](#)
- [System.Net.CoreResponseData.m_ResponseHeaders field](#)
- [System.Net.CoreResponseData.m_StatusCode field](#)
- [System.Net.ExceptionHelper class](#)
- [System.Net.HttpStatusDescription class](#)
- [System.Net.HttpWebRequest._AutoRedirects field](#)
- [System.Net.HttpWebRequest._CoreResponse field](#)
- [System.Net.HttpWebRequest._HttpResponse field](#)
- [System.Net.Logging class](#)
- [System.Net.Mail.MailAddressParser class](#)
- [System.Net.Mail.QuotedPairReader class](#)

- System.Net.Mime.MailBnfHelper class
- System.Net.PooledStream.NetworkStream property
- System.Net.RtcState class
- System.Net.Security.SslState.SslProtocol property
- System.Net.ServicePoint.m_ConnectionGroupList field
- System.Net.ServicePointManager.CloseConnectionGroups method
- System.Net.ServicePointManager.s_ServicePointTable field
- System.Net.TlsStream.m_Worker field
- System.Net.UnsafeNclNativeMethods class
- System.Net.WebHeaderCollection.AddInternal method
- System.ServiceModel.Channels.Message.BodyToString method
- System.ServiceModel.Channels.Message.WriteStartHeaders method
- System.Web.Compilation.ControlBuilderInterceptor class
- System.Windows.Controls.GridViewHeaderRowPresenter.FindHeaderByColumn method
- System.Windows.Controls.GridViewHeaderRowPresenter.MakeParentItemsControlIGotFocus method
- System.Windows.Controls.GridViewHeaderRowPresenter.PrepareHeaderDrag method
- System.Windows.Diagnostics.VisualDiagnostics.s_isDebuggerCheckDisabledForTestPurposes field
- System.Windows.Forms.ContainerControl.ResetActiveAndFocusedControlsRecursive method
- System.Windows.Forms.Control.UpdateStylesCore method
- System.Windows.Forms.ControlPaint.CalculateBackgroundImageRectangle method
- System.Windows.Forms.Design.DataMemberFieldEditor class
- System.Windows.Forms.Design.DataMemberListEditor class
- System.Windows.Forms.Design.StringCollectionEditor class
- System.Xml.XmlReader.CreateSqlReader method
- adodb.Connection interface
- adodb.EventReason enum
- adodb.EventStatus enum
- stdole.DISPPARAMS Structure
- stdole.EXCEPINFO Structure
- stdole.IFont.Name property
- stdole.IFontDisp interface
- stdole.IPicture.Handle property
- stdole.IPictureDisp.Handle property
- stdole.StdFont interface
- stdole.StdPicture interface

See also

- [.NET Framework and Out-of-Band Releases](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SmiOrderProperty.Item Property

Article • 09/15/2021

Gets the column order for the entity. The assembly that contains this property has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

Syntax

C#

```
internal SmiColumnOrder Item { get; }
```

Property value

The column order.

Remarks

⚠ Warning

The `SmiOrderProperty.Item` property is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [Microsoft.SqlServer.Server](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Exception.PrepForRemoting Method

Article • 09/15/2021

Preserves the server-side stack trace by appending it to the message before the exception is rethrown at the client call site.

C#

```
internal Exception PrepForRemoting();
```

Returns

[Exception](#)

This [Exception](#) instance.

Remarks

Warning

The `Exception.PrepForRemoting` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System](#)

Assembly: mscorlib.dll (in mscorlib.dll)

.NET Framework versions: Available since 1.0.

SqlBinary Constructor

Article • 04/07/2022

Initializes a new instance of the `SqlBinary` struct without making a copy of the `byte` array passed via the `value` parameter. Future mutations to this array may cause the resulting `SqlBinary` instance to change its value.

The `ignored` parameter is ignored.

C#

```
internal SqlBinary(byte[] value, bool ignored);
```

Remarks

⚠ Warning

This overload of the `SqlBinary` constructor is internal and is not meant to be used directly in your code. This API may not be available in future versions of .NET.

Microsoft does not support the use of this constructor in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

 .NET

 .NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

more information, see [our contributor guide](#).

 [Provide product feedback](#)

SqlChars.Stream Property

Article • 09/15/2021

Gets or sets the character stream. The assembly that contains this property has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
internal SqlStreamChars Stream { get; set; }
```

Property value

`System.Data.SqlTypes.SqlStreamChars`

The character stream.

Remarks

⚠ Warning

The `SqlChars.Stream` property is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

SqlGuid Constructor

Article • 04/07/2022

Initializes a new instance of the `SqlGuid` struct without making a copy of the `byte` array passed via the `value` parameter. Future mutations to this array may cause the resulting `SqlGuid` instance to change its value.

The `ignored` parameter is ignored.

C#

```
internal SqlGuid(byte[] value, bool ignored);
```

Remarks

⚠ Warning

This overload of the `SqlGuid` constructor is internal and is not meant to be used directly in your code. This API may not be available in future versions of .NET.

Microsoft does not support the use of this constructor in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

 .NET

 .NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

more information, see [our contributor guide](#).

 [Provide product feedback](#)

SqlMoney Constructor

Article • 10/11/2022

Initializes a new instance of the `SqlMoney` struct, where `value` has already been scaled by a ten-thousandth of a currency unit. For example, if `20000` is provided for the `value` parameter, this `SqlMoney` instance will represent `2` currency units.

The `ignored` parameter is ignored.

C#

```
internal SqlMoney(long value, int ignored);
```

Remarks

⚠ Warning

This overload of the `SqlMoney` constructor is internal and is not meant to be used directly in your code. This API may not be available in future versions of .NET.

Microsoft does not support the use of this constructor in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

 .NET

 .NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

more information, see [our contributor guide](#).

 [Provide product feedback](#)

SqlMoney.ToSqlInternalRepresentation Method

Article • 10/11/2022

Returns the value of this `SqlMoney` instance scaled by a ten-thousandth of a currency unit. For example, if the current `SqlMoney` instance represents 2 currency units, the `ToSqlInternalRepresentation` method will return 20000.

If this `SqlMoney` instance represents a null value (see `IsNull`), calling this method will throw a `SqlNullValueException`.

C#

```
internal long ToSqlInternalRepresentation();
```

Remarks

⚠ Warning

This method is internal and is not meant to be used directly in your code. This API may not be available in future versions of .NET.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars Constructor

Article • 09/15/2021

Initializes a new instance of the `SqlStreamChars` class. The assembly that contains this constructor has a friend relationship with `SQLAccess.dll`. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
protected SqlStreamChars();
```

Remarks

⚠ Warning

The `SqlStreamChars` constructor is protected and is not meant to be used directly in your code.

Microsoft does not support the use of this constructor in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in `System.Data.dll`)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
[GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.CanSeek Property

Article • 09/15/2021

When overridden in a derived class, gets a value that indicates whether the current stream supports the seek operation. The assembly that contains this property has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public abstract bool CanSeek { get; }
```

Property value

Boolean

true if the current stream supports the seek operation; otherwise, false.

Remarks

⚠ Warning

The `SqlStreamChars.CanSeek` property is private and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.



Collaborate with us on



.NET feedback

GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.IsNull Property

Article • 09/15/2021

When overridden in a derived class, gets a value that indicates whether the stream is `null`. The assembly that contains this property has a friend relationship with `SQLAccess.dll`. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

Syntax

C#

```
public abstract bool IsNull { get; }
```

Property value

Boolean

`true` if the stream is `null`; otherwise, `false`.

Remarks

⚠ Warning

The `SqlStreamChars.IsNull` property is private and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: `System.Data` (in `System.Data.dll`)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.Length Property

Article • 09/15/2021

When overridden in a derived class, gets the length of the current stream. The assembly that contains this property has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

Syntax

C#

```
public abstract long Length { get; }
```

Property value

[Int64](#)

The length of the stream.

Remarks

Warning

The `SqlStreamChars.Length` property is private and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.Close Method

Article • 09/15/2021

Closes the current stream and releases any system resources associated with the stream. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public virtual void Close();
```

Remarks

Warning

The `SqlStreamChars.Close` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

more information, see [our contributor guide](#).

SqlStreamChars.Dispose(Boolean) Method

Article • 09/15/2021

When overridden in a derived class, releases the resources used by the stream. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
protected virtual void Dispose (bool disposing);
```

Parameters

`disposing`

`true` to release both managed and unmanaged resources; `false` to release only unmanaged resources.

Remarks

⚠ Warning

The `SqlStreamChars.Dispose` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.Flush Method

Article • 09/15/2021

When overridden in a derived class, clears all buffers for this stream and causes any buffered data to be written to the underlying device. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

Syntax

C#

```
public abstract void Flush();
```

Remarks

Warning

The `SqlStreamChars.Flush` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

issues and pull requests. For more information, see [our contributor guide](#).

 Open a documentation issue

 Provide product feedback

SqlStreamChars.Read(Char[], Int32, Int32) Method

Article • 09/15/2021

When overridden in a derived class, reads the next set of characters from the input stream. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public abstract int Read (char[] buffer, int offset, int count);
```

Parameters

`buffer`

A char array to read.

`offset`

An offset relative to origin.

`count`

The number of characters to be read from the current stream.

Returns

`Int32`

The total number of characters read into the buffer.

Remarks

⚠ Warning

The `SqlStreamChars.Read` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.Seek(Int64, SeekOrigin) Method

Article • 09/15/2021

When overridden in a derived class, sets the position within the current stream. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public abstract long Seek (long offset, System.IO.SeekOrigin origin);
```

Parameters

`offset`

A byte offset relative to `origin`.

`origin`

One of the enumeration values that indicates the reference point from which to obtain the new position.

Returns

`Int32`

The new position within the current stream.

Remarks

Warning

The `SqlStreamChars.Seek` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.SetLength(Int64) Method

Article • 09/15/2021

When overridden in a derived class, releases the resources used by the stream. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public abstract void SetLength (long value);
```

Parameters

`value`

The desired length of the current stream in bytes.

Remarks

Warning

The `SqlStreamChars.SetLength` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

SqlStreamChars.Write(Char[], Int32, Int32) Method

Article • 09/15/2021

When overridden in a derived class, writes a sequence of characters to the current stream and advances the current position within this stream by the number of characters written. The assembly that contains this method has a friend relationship with SQLAccess.dll. It's intended for use by SQL Server. For other databases, use the hosting mechanism provided by that database.

C#

```
public abstract void Write (char[] buffer, int offset, int count);
```

Parameters

`buffer` A char array to write.

`offset` An offset relative to origin.

`count` The number of characters to be written to the current stream.

Remarks

⚠ Warning

The `SqlStreamChars.Write` method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method write in a production application under any circumstance.

Requirements

Namespace: [System.Data.SqlTypes](#)

Assembly: System.Data (in System.Data.dll)

.NET Framework versions: Available since 2.0.

 **Collaborate with us on GitHub**

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

MemoryStream.InternalGetOriginAndLength method

Article • 09/15/2021

Gets the internal values of origin and length of the memory stream.

C#

```
internal void InternalGetOriginAndLength(out int origin, out int length)
```

Parameters

- `origin` `Int32`

When this method returns, the offset of the byte array specified when creating a new `MemoryStream` object. Contains 0 if the byte array was created by `MemoryStream`.

- `length` `Int32`

When this method returns, the number of bytes within the memory stream.

Remarks

⚠ Warning

The `MemoryStream.InternalGetOriginAndLength` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.IO](#)

Assembly: mscorelib.dll (in mscorelib.dll)

.NET Framework versions: Available since 2.0.

ComNetOS class

Article • 09/15/2021

Provides information about the current operating system, such as its version and installation type (client or server). This class cannot be inherited.

C#

```
internal static class ComNetOS
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

IsWin7orLater field

Specifies whether the operating system version is Windows 7 or later.

C#

```
internal static readonly bool IsWin7orLater
```

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

more information, see [our contributor guide](#).

 [Provide product feedback](#)

Connection Class

Article • 09/15/2021

The `Connection` class parses server responses, queue requests, and pipeline requests.

Syntax

C#

```
internal class Connection : PooledStream
```

⚠ Warning

The `Connection` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Connection.m_WriteList Field

Article • 09/15/2021

`Connection.m_WriteList` is an [ArrayList](#) of [HttpWebRequest](#) objects that are queued up to be sent over HTTP.

Syntax

C#

```
private ArrayList m_WriteList
```

Warning

The `Connection.m_WriteList` field is private and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

ConnectionGroup Class

Article • 09/15/2021

The `ConnectionGroup` class groups a list of connections within the `ServicePoint` context and is used to maintain context for network resources (for example, proxies and separate clients).

Syntax

C#

```
internal class ConnectionGroup
```

⚠ Warning

The `ConnectionGroup` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

ConnectionGroup.m_ConnectionList Field

Article • 09/15/2021

`ConnectionGroup.m_ConnectionList` is an [ArrayList](#) of connection objects that serves the same URI and share the same values for some other properties like expiration and authentication.

Syntax

C#

```
private ArrayList m_ConnectionList
```

⚠ Warning

The `ConnectionGroup.m_ConnectionList` field is private and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

ConnectStream.Connection Property

Article • 09/15/2021

Gets the connection.

Syntax

C#

```
internal Connection Connection { get; }
```

Property value

`System.Net.Connection`

The connection object.

Remarks

Warning

The `ConnectStream.Connection` property is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 1.0.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

CoreResponseData Class

Article • 09/15/2021

The `CoreResponseData` class represents the parsing of the HTTP headers and the response body.

Syntax

C#

```
internal class CoreResponseData
```

Warning

This API is internal, and it is not meant to be used directly in your code. Instead, you should use a [DiagnosticSource](#) to hook networking code. See [DiagnosticSource User's Guide](#).

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

CoreResponseData.m_ResponseHeaders Field

Article • 09/15/2021

`CoreResponseData.m_ResponseHeaders` is a [WebHeaderCollection](#) of headers associated with the server response.

Syntax

C#

```
public WebHeaderCollection m_ResponseHeaders
```

⚠ Warning

This API is not meant to be used directly in your code. Instead, you should use a [DiagnosticSource](#) to hook networking code. See [DiagnosticSource User's Guide ↗](#).

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

CoreResponseData.m_StatusCode Field

Article • 09/15/2021

`CoreResponseData.m_StatusCode` is an [HttpStatusCode](#) containing the status of the response.

Syntax

C#

```
public HttpStatusCode m_StatusCode
```

Warning

This API is not meant to be used directly in your code. Instead, you should use a [DiagnosticSource](#) to hook networking code. See [DiagnosticSource User's Guide](#).

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

ExceptionHelper class

Article • 09/15/2021

Provides exceptions with standardized error messages. This class cannot be inherited.

C#

```
internal static class ExceptionHelper
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

WebPermissionUnrestricted field

Specifies that the app can connect to internet resources.

C#

```
internal static readonly WebPermission WebPermissionUnrestricted
```

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

⌚ Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

⌚ Open a documentation issue

👤 Provide product feedback

more information, see [our contributor guide](#).

HttpStatusDescription class

Article • 09/15/2021

Provides standard HTTP status descriptions. This class cannot be inherited.

C#

```
internal static class HttpStatusDescription
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Get method

Returns the description associated with the specified HTTP status code.

C#

```
internal static string Get(int code)
```

Parameters

code [Int32](#)

The HTTP status code, for example, [404](#).

Return value

[System.String](#)

The HTTP status description.

Requirements

Namespace: [System.Net](#)

 **Collaborate with us on GitHub**

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

 **.NET feedback**

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

HttpWebRequest._AutoRedirects Field

Article • 09/15/2021

`HttpWebRequest._AutoRedirects` is an `Int32` that reflects the number of redirects made for this `HttpWebRequest`.

Syntax

C#

```
private int _AutoRedirects
```

⚠ Warning

The `HttpWebRequest._AutoRedirects` field is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

HttpWebRequest._CoreResponse Field

Article • 09/15/2021

`HttpWebRequest._CoreResponse` is an object (either a `CoreresponseData` or an `Exception`) containing the result of HTTP response parsing.

Syntax

C#

```
private object _CoreResponse
```

⚠ Warning

This API is not meant to be used directly in your code. Instead, you should use a `DiagnosticSource` to hook networking code. See [DiagnosticSource User's Guide](#).

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

HttpWebRequest._HttpResponse Field

Article • 09/15/2021

`HttpWebRequest._HttpResponse` is an [HttpWebResponse](#) containing HTTP response details from an HTTP request. It can be `null` until an HTTP response is received.

Syntax

C#

```
internal HttpWebResponse _HttpResponse
```

⚠ Warning

The `HttpWebRequest._HttpResponse` field is internal and not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

Logging class

Article • 09/15/2021

Provides trace logging functionality.

C#

```
internal class Logging
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Associate method

Logs information that two objects are associated with each other.

C#

```
internal static void Associate(TraceSource traceSource, object objA, object objB)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `objA` [Object](#)

The object to associate with `objB`.

- `objB` [Object](#)

The object to associate with `objA`.

Enter(TraceSource, object, string, string) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, object obj, string  
method, string param)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [Object](#)

The object that the method was called on.

- `method` [String](#)

The method that is being entered.

- `param` [String](#)

The parameters that were passed to the method.

Enter(TraceSource, object, string, object) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, object obj, string  
method, object paramObject)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [Object](#)

The object that the method was called on.

- `method` [String](#)

The method that is being entered.

- `paramObject` [Object](#)

The parameters that were passed to the method.

Enter(TraceSource, string, string, string) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, string obj, string  
method, string param)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [String](#)

The object that the method was called on.

- `method` [String](#)

The method that is being entered.

- `param` [String](#)

The parameters that were passed to the method.

Enter(TraceSource, string, string, object) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, string obj, string
method, object paramObject)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [String](#)

The object that the method was called on.

- `method` [String](#)

The method that is being entered.

- `paramObject` [Object](#)

The parameters that were passed to the method.

Enter(TraceSource, string, string) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, string method, string
parameters)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `method` `String`

The method that is being entered.

- `parameters` `String`

The parameters that were passed to the method.

Enter(TraceSource, string) method

Logs entrance to a method.

C#

```
internal static void Enter(TraceSource traceSource, string msg)
```

Parameters

- `traceSource` `TraceSource`

The trace source to log the event to.

- `msg` `String`

The entrance message to log to the trace source.

Exception method

Logs an exception and restores indentation.

C#

```
internal static void Exception(TraceSource traceSource, object obj, string  
method, Exception e)
```

Parameters

- `traceSource` `TraceSource`

The trace source to log the event to.

- `obj` `Object`

The object that the method that threw an exception was called on.

- `method` [String](#)

The method that threw the exception.

- `e` [Exception](#)

The exception that was thrown.

Exit(TraceSource, object, string, object) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, object obj, string  
method, object retObject)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [Object](#)

The object that the method was called on.

- `method` [String](#)

The method that is being exited.

- `retObject` [Object](#)

The value that's being returned by the method.

Exit(TraceSource, string, string, object) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, string obj, string
```

```
method, object retObject)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [String](#)

The object that the method was called on.

- `method` [String](#)

The method that is being exited.

- `retObject` [Object](#)

The value that's being returned by the method.

Exit(TraceSource, object, string, string) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, object obj, string  
method, string retValue)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [Object](#)

The object that the method was called on.

- `method` [String](#)

The method that is being exited.

- `retValue` [String](#)

The value that's being returned by the method.

Exit(TraceSource, string, string, string) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, string obj, string
method, string retValue)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `obj` [String](#)

The object that the method was called on.

- `method` [String](#)

The method that is being exited.

- `retValue` [String](#)

The value that's being returned by the method.

Exit(TraceSource, string, string) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, string method, string
parameters)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `method` [String](#)

The method that is being exited.

- `parameters` [String](#)

The parameters that were passed to the method that's being exited.

Exit(TraceSource, string) method

Logs exit from a function.

C#

```
internal static void Exit(TraceSource traceSource, string msg)
```

Parameters

- `traceSource` [TraceSource](#)

The trace source to log the event to.

- `msg` [String](#)

The exit message to log to the trace source.

Http property

Gets the trace source for "System.Net.Http".

C#

```
internal static TraceSource Http { get; }
```

Property value

[TraceSource](#)

The trace source for "System.Net.Http", or `null` if logging is not enabled.

On property

Gets a value that indicates whether logging is enabled.

C#

```
internal static bool On { get; }
```

Property value

Boolean

`true` if logging is enabled; otherwise, `false`.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

MailAddressParser class

Article • 09/15/2021

Parses email addresses as described in RFC 2822. This class cannot be inherited.

C#

```
internal static class MailAddressParser
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

ParseAddress method

Parses a single email address from the specified string.

C#

```
internal static MailAddress ParseAddress(string data)
```

Parameters

`data` [String](#)

The string that contains an email address to be parsed.

Return value

[MailAddress](#)

A valid email address.

Exceptions

[System.FormatException](#)

The email address is invalid.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

MailBnfHelper class

Article • 09/15/2021

Contains utility methods for parsing internet message-formatted strings. This class cannot be inherited.

C#

```
internal static class MailBnfHelper
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Ascii7bit.MaxValue field

Represents the maximum 7-bit Ascii value.

C#

```
internal static readonly int Ascii7bit.MaxValue
```

Atext field

Represents the characters allowed in atoms.

C#

```
internal static bool[] Atext
```

CR field

Represents the carriage-return character.

C#

```
internal static readonly char CR
```

Ctext field

Represents the characters allowed inside of comments.

```
C#
```

```
internal static bool[] Ctext
```

Dot field

Represents the full-stop character (.).

```
C#
```

```
internal static readonly char Dot
```

EndComment field

Represents the character that specifies the end of a comment.

```
C#
```

```
internal static readonly char EndComment
```

LF field

Represents the line-feed character.

```
C#
```

```
internal static readonly char LF
```

Space field

Represents the space character.

C#

```
internal static readonly char Space
```

StartComment field

Represents the character that specifies the start of a comment.

C#

```
internal static readonly char StartComment
```

Tab field

Represents the tab character.

C#

```
internal static readonly char Tab
```

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PooledStream.NetworkStream Property

Article • 09/15/2021

Gets or sets the network stream for the `PooledStream` socket.

Syntax

C#

```
internal NetworkStream NetworkStream { get; set; }
```

Property value

`NetworkStream` The network stream for the `PooledStream` socket.

Remarks

⚠ Warning

The `PooledStream.NetworkStream` property is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

QuotedPairReader class

Article • 09/15/2021

Determines which characters are quoted (escaped) in a quoted string. This class cannot be inherited.

C#

```
internal static class QuotedPairReader
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

CountQuotedChars method

Counts the number of consecutive quoted characters, including multiple preceding quoted backslashes, in the specified string. For example, given the string `a\\b` and an index of `4`, the method returns `4`, because `b` is quoted and so are the three preceding backslashes.

C#

```
internal static int CountQuotedChars(string data, int index, bool permitUnicodeEscaping)
```

Parameters

- `data` `String`

The data string in which to count consecutive quoted characters.

- `index` `Int32`

The position in the specified string up to and including which consecutive quoted characters should be counted.

- `permitUnicodeEscaping Boolean`

`true` to permit Unicode characters to be escaped; otherwise, `false`.

Return value

`System.Int32`

`0` if the character at the specified index is not escaped; otherwise, the number of consecutive quoted characters up to and including the character at `index`.

Exceptions

`System.FormatException`

An escaped Unicode character was found but is not permitted.

Requirements

Namespace: `System.Net`

Assembly: `System` (in `System.dll`)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

RtcState Class

Article • 09/15/2021

The `RtcState` class represents state data for a real-time communication (RTC) request.

Syntax

C#

```
internal class RtcState
```

⚠ Warning

The `RtcState` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 4.5.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

ServicePoint.m_ConnectionGroupList Field

Article • 09/15/2021

`ServicePoint.m_ConnectionGroupList` is a [Hashtable](#) of connection groups, each holding a connection for the [ServicePoint](#)'s URI.

Syntax

C#

```
private Hashtable m_ConnectionGroupList
```

⚠ Warning

The `ServicePoint.m_ConnectionGroupList` field is private and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

ServicePointManager.CloseConnectionGroups method

Article • 09/15/2021

Iterates through all service points and closes connection groups that have the specified name.

C#

```
internal static void CloseConnectionGroups(string connectionGroupName)
```

⚠ Warning

This method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Parameters

connectionGroupName [String](#)

The name of the connection group to close.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

more information, see [our contributor guide](#).

ServicePointManager.s_ServicePointTable Field

Article • 09/15/2021

`ServicePointManager.s_ServicePointTable` is a [Hashtable](#) that contains the list of active HTTP connections ([ServicePoints](#)) in the [AppDomain](#).

Syntax

C#

```
private static Hashtable s_ServicePointTable
```

⚠ Warning

The `ServicePointManager.s_ServicePointTable` field is private and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

TlsStream.m_Worker Field

Article • 09/15/2021

Represents the state of the SSL stream.

Syntax

C#

```
private SslState m_Worker;
```

Field value

`System.Net.Security.SslState` The state of the SSL stream.

Remarks

⚠ Warning

The `TlsStream.m_Worker` field is private and is not meant to be used directly in your code.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

UnsafeNclNativeMethods class

Article • 09/15/2021

Contains classes that import unsafe native networking methods. This class cannot be inherited.

C#

```
internal static class UnsafeNclNativeMethods
```

⚠ Warning

This class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

NativePKI class

Contains methods imported from crypt32.dll. These methods handle certificates when using Hypertext Transfer Protocol Secure (HTTPS). This class cannot be inherited.

C#

```
internal static class NativePKI
```

NativePKI.FindClientCertificates method

Discovers available client certificates to send to the server.

C#

```
internal static X509CertificateCollection FindClientCertificates
```

Return value

[System.Security.Cryptography.X509Certificates.X509CertificateCollection](#)

A collection of available client certificates.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

WebHeaderCollection.AddInternal method

Article • 09/15/2021

Adds a new header with the specified name and value to the collection, bypassing checks.

C#

```
internal void AddInternal(string name, string value)
```

⚠ Warning

This method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Parameters

- `name` [String](#)

The name of the header.

- `value` [String](#)

The value of the header.

Requirements

Namespace: [System.Net](#)

Assembly: System (in System.dll)



Collaborate with us on
GitHub



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 [Open a documentation issue](#)

 [Provide product feedback](#)

SslState.SslProtocol Property

Article • 09/15/2021

Gets the SSL protocol versions.

Syntax

C#

```
internal SslProtocols SslProtocol { get; }
```

Property value

[SslProtocols](#) A bitwise combination of the enumeration values that specify the SSL protocol versions.

Remarks

⚠ Warning

The `SslState.SslProtocol` property is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this property in a production application under any circumstance.

Requirements

Namespace: [System.Net.Security](#)

Assembly: System (in System.dll)

.NET Framework versions: Available since 2.0.



Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

Message.BodyToString Method

Article • 09/15/2021

Converts the message body into a string by calling the [Message.OnBodyToString](#) method.

C#

```
internal void BodyToString(XmlDictionaryWriter writer);
```

Parameters

- `writer` [XmlDictionaryWriter](#)

The writer that is used to convert the message body to a string.

Remarks

⚠ Warning

The `Message.BodyToString` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.ServiceModel.Channels](#)

Assembly: System.ServiceModel.dll

.NET Framework versions: Available since 3.0.

Message.WriteStartHeaders Method

Article • 09/15/2021

Writes the start header into an XML file by calling the [Message.OnWriteStartHeaders](#) method.

C#

```
internal void WriteStartHeaders(XmlDictionaryWriter writer)
```

Parameters

- `writer` [XmlDictionaryWriter](#)

The writer that is used to write the start header into an XML file.

Remarks

⚠ Warning

The `Message.WriteStartHeaders` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.ServiceModel.Channels](#)

Assembly: System.ServiceModel.dll

.NET Framework versions: Available since 3.0.

ControlBuilderInterceptor class

Article • 09/15/2021

The `ControlBuilderInterceptor` class allows the compilation process to be customized or controlled.

Syntax

C#

```
internal class ControlBuilderInterceptor
```

⚠ Warning

The `ControlBuilderInterceptor` class is internal and is not meant to be used directly in your code.

As described in the Remarks section, the existence of this type can be checked to determine whether interceptor type support is present. Microsoft does not support any other use of this class in a production application under any circumstance.

Remarks

In .NET Framework 2.0 and .NET Framework 3.5, the [August 2020](#) updates added support for using an interceptor type to customize or control the compilation process. You can determine whether this support is present by using `Type.GetType()` to check the existence of the `ControlBuilderInterceptor` type, as demonstrated in the following code.

C#

```
Type type = Type.GetType("System.Web.Compilation.ControlBuilderInterceptor,
System.Web, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a");
```

If the return value is non-null, then interceptor support is present. If the return value is `null`, or if an exception is thrown, then the August 2020 updates have not been installed, and interceptor support is absent.

If interceptor support is present, you can write and register an interceptor type that will interact with the compilation process in exactly the same way that [ControlBuilderInterceptor](#) does on later versions of .NET Framework. In .NET Framework 2.0 and .NET Framework 3.5, the interceptor type can be any class that meets the following requirements:

- Has a public, parameterless constructor.
- Has public, non-static methods named `PreControlBuilderInit` and `OnProcessGeneratedCode` that have the same signature and semantics as the `PreControlBuilderInit(ControlBuilder, TemplateParser, ControlBuilder, Type, String, String, IDictionary, IDictionary)` and `OnProcessGeneratedCode(ControlBuilder, CodeCompileUnit, CodeTypeDeclaration, CodeTypeDeclaration, CodeMemberMethod, CodeMemberMethod, IDictionary)` methods, which exist in later versions of .NET Framework.

Register the interceptor type by using the `aspnet:20ControlBuilderInterceptor` key in ASP.NET application settings (`<appSettings>`). This application setting must be listed in your computer or application `web.config` file. Specify the interceptor type by using its assembly-qualified type name. The following example shows how to register an interceptor type named `Fabrikam.Interceptor`.

XML

```
<configuration>
  ...
  <appSettings>
    ...
    <add key="aspnet:20ControlBuilderInterceptor"
        value="Fabrikam.Interceptor, Fabrikam, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=2b3831f2f2b744f7" />
  </appSettings>
</configuration>
```

To retrieve the assembly-qualified name of a type, use the `Type.AssemblyQualifiedName` property, as demonstrated in the following code.

C#

```
string assemblyQualifiedName =
typeof(Fabrikam.Interceptor).AssemblyQualifiedName;
```

When interceptor support is present, the compilation process interacts with the listed type in the manner described above. When interceptor support is absent, the application setting is ignored and has no effect.

Requirements

Namespace: System.Web.Compilation

Assembly: System.Web (in System.Web.dll)

.NET Framework versions: 3.5, 2.0

FindHeaderByColumn method

Article • 09/15/2021

Finds the column header for the specified column in the visual tree.

C#

```
private GridViewColumnHeader FindHeaderByColumn(GridViewColumn column)
```

⚠ Warning

This method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Parameters

`column` [GridViewColumn](#)

The column whose header should be found and returned.

Return value

[GridViewColumnHeader](#)

The header of the specified column, or `null` if the specified column doesn't exist.

Requirements

Namespace: [System.Windows.Controls](#)

Assembly: PresentationFramework.dll

See also

- [GridViewHeaderRowPresenter](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

MakeParentItemsControlGotFocus method

Article • 09/15/2021

Gives focus to the parent control of the item. If the parent control is a [ListBox](#), gives focus to the most recently accessed item in the [ListBox](#).

C#

```
internal void MakeParentItemsControlGotFocus()
```

⚠ Warning

This method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Controls](#)

Assembly: PresentationFramework.dll

See also

- [GridViewHeaderRowPresenter](#)

 Collaborate with us on
[GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

PrepareHeaderDrag method

Article • 09/15/2021

Prepares the specified column header for reordering.

C#

```
private void PrepareHeaderDrag(GridViewColumnHeader header, Point pos, Point relativePos, bool cancelInvoke)
```

⚠ Warning

This method is private and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Parameters

`header` [GridViewColumnHeader](#)

The column header to prepare for reordering.

`pos` [Point](#)

The position, relative to [GridViewHeaderRowPresenter](#), where the dragging starts.

`relativePos` [Point](#)

The position, relative to `header`, where the dragging starts.

`cancelInvoke` [Boolean](#)

`true` to cancel the reordering; otherwise, `false`.

Requirements

Namespace: [System.Windows.Controls](#)

Assembly: PresentationFramework.dll

See also

- [GridViewHeaderRowPresenter](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

s_isDebuggerCheckDisabledForTestPurposes Field

Article • 09/15/2021

This private field in the `System.Windows.Diagnostics.VisualDiagnostics` class is used by Visual Studio to determine whether an internal check for an active debugger will be performed.

Syntax

C#

```
private static bool s_isDebuggerCheckDisabledForTestPurposes
```

⚠ Warning

APIs in the `System.Windows.Diagnostics.VisualDiagnostics` class are only available when an application is running under a debugger. Set `s_isDebuggerCheckDisabledForTestPurposes` to `true` to access the APIs outside of a debugger.

Microsoft does not support the use of this field in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Diagnostics](#)

Assembly: PresentationCore (in PresentationCore.dll)

.NET Framework versions: Available since 4.6.

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 Open a documentation issue

 Provide product feedback

ResetActiveAndFocusedControlsRecursive method

Article • 11/13/2021

Resets the active control and the control that has focus. If the active control is itself a container control, this method is called on the active control before it's reset.

Syntax

C#

```
internal void ResetActiveAndFocusedControlsRecursive()
```

⚠ Warning

The `ResetActiveAndFocusedControlsRecursive` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms](#)

Assembly: System.Windows.Forms (in System.Windows.Forms.dll)

.NET Framework versions: Available since 1.0.

See also

- [ContainerControl](#)

 Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

CalculateBackgroundImageRectangle method

Article • 11/13/2021

Calculates the location and size for a background image with a specified layout, for example, stretched, centered, or zoomed.

Syntax

C#

```
internal static Rectangle CalculateBackgroundImageRectangle(Rectangle  
bounds, Image backgroundImage, ImageLayout imageLayout)
```

Parameters

`bounds`

The client rectangle for `backgroundImage`.

`backgroundImage`

The image whose location and size is to be calculated. If `null`, the client rectangle `bounds` is returned.

`imageLayout`

One of the enumeration values that specifies the position of the background image relative to `bounds`.

Returns

`Rectangle`

The location and size for the specified background image with the specified layout.

⚠ Warning

The `CalculateBackgroundImageRectangle` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms](#)

Assembly: System.Windows.Forms (in System.Windows.Forms.dll)

.NET Framework versions: Available since 1.0.

See also

- [ControlPaint](#)

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

UpdateStylesCore method

Article • 11/13/2021

Forces the assigned styles to be reapplied to the control.

Syntax

C#

```
internal virtual void UpdateStylesCore()
```

⚠ Warning

The `UpdateStylesCore` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms](#)

Assembly: System.Windows.Forms (in System.Windows.Forms.dll)

.NET Framework versions: Available since 1.0.

See also

- [Control](#)
- [Control.UpdateStyles\(\)](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

issues and pull requests. For more information, see [our contributor guide](#).

 [Provide product feedback](#)

DataMemberFieldEditor Class

Article • 09/15/2021

Provides user interface for editing properties of data-bound objects by listing all properties of the `DataSource` object to select the value source from.

Syntax

C#

```
internal class DataMemberFieldEditor : UITypeEditor
```

⚠ Warning

The `DataMemberFieldEditor` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms.Design](#)

Assembly: System.Design (in System.Design.dll)

.NET Framework versions: Available since 2.0.

See also

- [System.Windows.Forms.Design](#)

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

issues and pull requests. For more information, see [our contributor guide](#).

 Open a documentation issue

 Provide product feedback

DataMemberListEditor Class

Article • 09/15/2021

Provides a drop-down user interface for editing properties of data-bound objects (objects that have non-null `DataSource` property) by listing all properties of the `DataSource` object to select the value source from.

Syntax

C#

```
internal class DataMemberListEditor : UITypeEditor
```

⚠ Warning

The `DataMemberListEditor` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this class in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms.Design](#)

Assembly: System.Design (in System.Design.dll)

.NET Framework versions: Available since 2.0.

See also

- [System.Windows.Forms.Design](#)

 Collaborate with us on
GitHub

The source for this content can
be found on GitHub, where you



.NET feedback

.NET is an open source project.
Select a link to provide feedback:

can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 Open a documentation issue

 Provide product feedback

StringCollectionEditor class

Article • 06/12/2023

Provides a user interface that can edit collections of strings at design time.

Syntax

C#

```
internal class StringCollectionEditor : CollectionEditor
```

⚠ Warning

The `StringCollectionEditor` class is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Windows.Forms](#)

Assembly: System.Design (in System.Design.dll)

.NET Framework versions: Available since 2.0.

 Collaborate with us on
[GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

XpsDocumentWriter.raise_WritingCancelled Method

Article • 09/15/2021

Raises the [WritingCancelled](#) event.

Syntax

C#

```
public void raise_WritingCancelled (object value0,
    System.Windows.Documents.Serialization.WritingCancelledEventArgs value1);
```

Parameters

- `value0 Object`
The source of the event.
- `value1 WritingCancelledEventArgs`
The event data.

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: 3.0

XpsDocumentWriter.raise_WritingCompleted Method

Article • 09/15/2021

Raises the [WritingCompleted](#) event.

Syntax

C#

```
public void raise_WritingCompleted (object value0,
    System.Windows.Documents.Serialization.WritingCompletedEventArgs value1);
```

Parameters

- `value0 Object`
The source of the event.
- `value1 WritingCompletedEventArgs`
The event data.

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: 3.0

XpsDocumentWriter.raise_WritingPrintTicketRequired Method

Article • 09/15/2021

Raises the [WritingPrintTicketRequired](#) event.

Syntax

C#

```
public void raise_WritingPrintTicketRequired (object value0,
    System.Windows.Documents.Serialization.WritingPrintTicketRequiredEventArgs
    value1);
```

Parameters

- `value0 Object`
The source of the event.
- `value1 WritingPrintTicketRequiredEventArgs`
The event data.

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: 3.0

XpsDocumentWriter.raise_WritingProgressChanged Method

Article • 09/15/2021

Raises the [WritingProgressChanged](#) event.

Syntax

C#

```
public void raise_WritingProgressChanged (object value0,
    System.Windows.Documents.Serialization.WritingProgressChangedEventArgs
    value1);
```

Parameters

- `value0 Object`
The source of the event.
- `value1 WritingProgressChangedEventArgs`
The event data.

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: 3.0

XpsDocumentWriter._WritingCancelled Event

Article • 09/15/2021

Occurs when a [Write](#) or [WriteAsync](#) operation is canceled.

Syntax

C#

```
internal event WritingCancelledEventHandler _WritingCancelled
```

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: Available since 3.0

XpsDocumentWriter._WritingCompleted Event

Article • 09/15/2021

Occurs when a write operation finishes.

Syntax

C#

```
internal event WritingCompletedEventHandler _WritingCompleted
```

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: Available since 3.0

XpsDocumentWriter._WritingProgressChanged Event

Article • 09/15/2021

Occurs when the [XpsDocumentWriter](#) updates its progress.

Syntax

C#

```
internal event WritingProgressChangedEventHandler _WritingProgressChanged
```

Requirements

Namespace: [System.Windows.Xps](#)

Assembly: System.Printing (in system.printing.dll)

.NET Framework versions: Available since 3.0

XmlReader.CreateSqlReader Method

Article • 09/15/2021

Creates a new [XmlReader](#) instance using the specified stream, settings, and context information for parsing.

C#

```
internal static XmlReader CreateSqlReader(Stream input,  
    XmlReaderSettings settings, XmlParserContext inputContext)
```

Parameters

- `input Stream`
The stream that contains the XML data.
- `settings XmlReaderSettings`
The settings for the new [XmlReader](#) instance. This value can be `null`.
- `inputContext XmlParserContext`
The context information required to parse the XML fragment. This value can be `null`.

Returns

[XmlReader](#)

An object that is used to read the XML data in the stream.

Remarks

⚠ Warning

The `XmlReader.CreateSqlReader` method is internal and is not meant to be used directly in your code.

Microsoft does not support the use of this method in a production application under any circumstance.

Requirements

Namespace: [System.Xml](#)

Assembly: System.Xml.dll

.NET Framework versions: Available since 2.0.

Connection Interface

Article • 09/15/2021

C#

```
[GuidAttribute("00000550-0000-0010-8000-00AA006D2EA4")]
public interface Connection : _Connection,
    ConnectionEvents_Event
```

Requirements

Namespace: adodb

Assembly: adodb (in adodb.dll)

EventReason Enumeration

Article • 09/15/2021

C#

```
[GuidAttribute("00000531-0000-0010-8000-00AA006D2EA4")]
public enum EventReasonEnum
```

Members

Member name	Description
adRsnAddNew	
adRsnDelete	
adRsnUpdate	
adRsnUndoUpdate	
adRsnUndoAddNew	
adRsnUndoDelete	
adRsnRequery	
adRsnResynch	
adRsnClose	
adRsnMove	
adRsnFirstChange	
adRsnMoveFirst	
adRsnMovePrevious	
adRsnMoveLast	

Requirements

Namespace: adodb

Assembly: adodb (in adodb.dll)

EventStatus Enumeration

Article • 09/15/2021

C#

```
[GuidAttribute("00000530-0000-0010-8000-00AA006D2EA4")]
public enum EventStatusEnum
```

Members

Member name	Description
adStatusOK	
adStatusErrorsOccurred	
adStatusCantDeny	
adRsnUndoUpdate	
adStatusCancel	
adStatusUnwantedEvent	

Requirements

Namespace: adodb

Assembly: adodb (in adodb.dll)

DISPPARAMS Structure

Article • 09/15/2021

C#

```
public struct DISPPARAMS
```

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

EXCEPINFO Structure

Article • 09/15/2021

C#

```
public struct EXCEPINFO
```

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

IFont.Name Property

Article • 09/15/2021

C#

```
string Name { get; set; }
```

Property value

[String](#)\

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

IFontDisp Interface

Article • 09/15/2021

C#

```
[GuidAttribute("BEF6E003-A874-101A-8BBA-00AA00300CAB")]
[InterfaceTypeAttribute()]
public interface IFontDisp
```

Requirements

Namespace: stdole

Assembly: stdole (in stdole.dll)

IPicture.Handle Property

Article • 09/15/2021

Gets the handle to the picture managed within this picture object to a specified location.

C#

```
int Handle { get; }
```

Property value

[Int32\](#)

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

IPictureDisp Interface

Article • 09/15/2021

C#

```
[InterfaceTypeAttribute()]
[GuidAttribute("7BF80981-BF32-101A-8BBB-00AA00300CAB")]
public interface IPictureDisp
```

Requirements

Namespace: stdole

Assembly: stdole (in stdole.dll)

IPictureDisp.Handle Property

Article • 09/15/2021

C#

```
int Handle { get; }
```

Property value

[Int32](#)\

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

StdFont Interface

Article • 09/15/2021

C#

```
[GuidAttribute("BEF6E003-A874-101A-8BBA-00AA00300CAB")]
public interface StdFont : Font,
    FontEvents_Event
```

Requirements

Namespace: stdole

Assembly: stdole (in stdole.dll)

StdPicture Interface

Article • 09/15/2021

C#

```
[GuidAttribute("7BF80981-BF32-101A-8BBB-00AA00300CAB")]
public interface StdPicture : Picture
```

Requirements

Namespace: `stdole`

Assembly: `stdole` (in `stdole.dll`)

POS for .NET v1.14.1 SDK Documentation

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) v1.14.1 is a class library that enables POS developers to apply Microsoft .NET technologies in their products.

POS for .NET v1.14.1 provides the following:

- A simple and consistent interface for .NET Framework applications to interact with POS devices.
- A set of interfaces and classes created to help vendors write applications for common devices such as Cash Drawers or Line Displays.
- A fully compliant implementation of the Unified Point of Service (UnifiedPOS) v1.14 international standard.
- Support for Windows Plug and Play functionality.

ⓘ Note

POS for .NET does not provide any support for the Windows Runtime (WinRT) API. For more information about POS device support in WinRT, see [Windows.Devices.PointOfService namespace](#) on MSDN.

The following documentation can help you use POS for .NET to develop POS applications and Service Objects (SOs) more efficiently, learn more about the benefits of POS for .NET, and achieve better compatibility with POS devices.

In This Section

- [Copyright Information for POS for .NET v1.14.1](#) Provides copyright information about POS for .NET v1.14.1.
- [POS for .NET FAQ](#) Provides answers to frequently asked questions for POS for .NET.
- [What's New in POS for .NET v1.14 and v1.14.1](#) Describes the features introduced in the 1.14 and 1.14.1 versions of POS for .NET.
- [POS for .NET v1.14.1 Features](#) Provides an in-depth view at POS for .NET, its architecture, and features in the following topics:
 - [POS for .NET Architecture](#)
 - [Service Object Overview](#)

- [Supported Device Classes](#)
- [Event Management](#)
- [POS Exception Handling](#)
- [POS for .NET Integration with Plug and Play](#)
- [Log Files](#)
- [Integration of Legacy Service Objects](#)
- **Developing a POS Application** Describes the POS for .NET predefined interfaces and classes, how best to use them in your code, and demonstrates suitable procedures for developing POS for .NET code in the following topics:
 - [Typical POS Application Architecture](#)
 - [POS for .NET API Support](#)
 - [Event Handler Sample](#)
- **Developing a Custom Service Object** Discusses the procedures, issues, and conventions for developing custom service objects in the following topics:
 - [POS for .NET Service Object Architecture](#)
 - [System Configuration](#)
 - [Service Object Samples: Getting Started](#)
 - [Developing Service Objects Using Base Classes](#)
 - [Device Input and Events](#)
 - [Device Output Models](#)
 - [Asynchronous Output Sample](#)
 - [Statistics Sample](#)
- [Base Class DirectIO Method](#)
- [Capability Properties](#)
- **Manually manage your POS for .NET devices** Describes how to manually manage your POS for .NET devices by modifying the POS for .NET configuration xml file.

- [POS Device Manager](#) Describes how POS for .NET helps you manage your POS devices in the following topics:
 - [Configure a device for remote management](#)
 - [Using the WMI API to Manage Devices](#)
 - [Using Visual Studio .NET Management Extensions and the POS for .NET WMI Management Classes](#)
 - [Using the POS Device Manager Command-Line Tool](#)

See Also

- [POS for .NET API Reference](#)

Reference

- [Microsoft.PointOfService](#)

Copyright Information for POS for .NET v1.14.1 (Microsoft Point of Service for .NET)

Article • 02/21/2023

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2017 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, JScript, MSDN, Visual Basic, Visual C#, Visual Studio, Windows, Windows Server, Windows Vista, and Win32 are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

POS for .NET FAQ (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Corporation

Contents

Installation

- How do I install POS for .NET 1.14 on my device?
- How do I install POS for .NET with my product?
- What platforms are supported for POS for .NET 1.14?

Device Types

- Which device types or categories does POS for .NET support?
- Which OPOS device types or categories does POS for .NET support?
- Which device simulators does POS for .NET include?

Service Objects

- How do I write a .NET service object? How do I get started?
- Where can I find the service object that I need for a specific device?

Installation

How do I install POS for .NET 1.14 on my device?

You can install Microsoft Point of Service for .NET (POS for .NET) v1.14 on your device by double clicking on the **POSforDotNet-1.14.msi** installer and following the installer wizard.

How do I install POS for .NET with my product?

You can silently run the POS for .NET v1.14 installer from within your own product installer by using the following commands:

To install the POS for .NET 1.14 runtime only

- Add the following command to your product installer:

```
msiexec /i POSforDotNet-1.14.msi /passive
```

To install the complete POS for .NET 1.14 including the SDK

1. Add the following command to your product installer:

```
msiexec /i POSforDotNet-1.14.msi INSTALLLEVEL=1000 /passive
```

What platforms are supported for POS for .NET 1.14?

POS for .NET v1.14 can be installed on the following platforms:

- Windows 11
- Windows 11 IoT Enterprise
- Windows 10
- Windows 10 IoT Enterprise
- Windows 10 IoT Enterprise LTSC 2021
- Windows 10 IoT Enterprise LTSC 2019

Device Types

Which device types or categories does POS for .NET support?

POS for .NET v1.14 supports all the point of service (POS) peripheral device categories defined in the Unified Point of Service (UnifiedPOS) v1.14 international standard.

Therefore, POS for .NET supports the following device categories:

- Belt
- Bill Acceptor
- Bill Dispenser
- Biometrics
- Bump Bar
- Cash Changer
- Cash Drawer
- Check Scanner
- Coin Acceptor

- Coin Dispenser
- Credit Authorization Terminal (CAT)
- Electronic Journal
- Electronic Value Reader/Writer
- Fiscal Printer
- Gate
- Hard Totals
- Image Scanner
- Item Dispenser
- Keylock
- Lights
- Line Display
- Magnetic Ink Character Recognition Reader (MICR)
- Magnetic Stripe Reader (MSR)
- Motion Sensor
- PIN Pad
- Point Card Reader/Writer
- POS Keyboard
- POS Power
- POS Printer
- Radio Frequency Identification (RFID) Scanner
- Remote Order Display
- Scale
- Scanner (Barcode Reader)
- Signature Capture
- Smart Card Reader/Writer
- Tone Indicator

Which OPOS device types or categories POS for .NET support?

POS for .NET fully supports any previously created OLE for Retail POS (OPOS) service objects for the following device categories:

- Bar Code Scanner
- Bump Bar
- Cash Changer
- Cash Drawer
- Check Scanner
- Coin Dispenser
- Credit Authorization Terminal (CAT)

- Fiscal Printer
- Hard Totals
- Keylock
- Line Display
- Magnetic Ink Character Recognition Reader (MICR)
- Magnetic Stripe Reader (MSR)
- Motion Sensor
- PIN Pad
- Point Card Reader/Writer
- POS Keyboard
- POS Power
- POS Printer
- Remote Order Display
- Scale
- Signature Capture
- Smart Card Reader/Writer
- Tone Indicator

Which device simulators does POS for .NET include?

When you install the POS for .NET Software Development Kit (SDK), it includes several device simulators. The simulators provide a simple means of simulating a device when no physical device is available. The simulators are helpful during the early stages of development, during prototyping, and for testing configurations before deployment.

POS for .NET includes simulators for the following device categories:

- Bar Code Scanner
- Cash Drawer
- Check Scanner
- Keylock
- Line Display
- Magnetic Stripe Reader (MSR)
- PIN Pad
- POS Keyboard
- POS Printer

Service Objects

How do I write a .NET service object? How do I get started?

It depends on the type of device that you want to support:

- For some devices, we offer base classes that implement most of the UnifiedPOS-specific functionality; therefore, you can focus on only the communications between the service object and the device.
- For other device categories, you can take advantage of our basic class for much of the functionality but implement certain aspects of the UnifiedPOS-specific functionality yourself.
- Finally, you can choose to do everything yourself. In this case, you implement the whole class based on a provided interface. To help you get started, POS for .NET SDK includes documentation and code for sample service objects.

Where can I find the service object that I need for a specific device?

You should contact the device manufacturer or your vendor to see whether they offer a .NET Service Object or a legacy OPOS service object for one of the supported legacy devices. Anyone can develop a .NET service object, and there are no requirements to register the service object with Microsoft. Therefore, we do not have a list available that identifies which devices are compatible with POS for .NET or OPOS.

What's New in POS for .NET v1.14 and v1.14.1 (Microsoft Point of Service for .NET)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) v1.14.1 has been updated to reflect that latest Unified Point of Service (UnifiedPOS) v1.14.1 international standard. In addition, POS for .NET v1.14 (and later) has been updated to work with .NET 4.0. POS for .NET v1.14.1 inherits all changes from v1.14.

Support for OLE for Retail POS (OPOS)

POS for .NET v1.14 (and later) provides support for the following additional OPOS service objects:

[\[+\] Expand table](#)

POS peripheral	Enumeration value
Fiscal printer	FISCALPRINTER
Smart card reader/writer	SMARTCARDRW
Bump bar	BUMPPBAR
Point card reader/writer	POINTCARDRW
Remote order display	REMOTEORDERDISPLAY
Cash changer	CASHCHANGER
Hard totals	HARDTOTALS
Motion sensor	MOTIONSENSOR

Changes in the POS for .NET 1.14.1 Release

POS for .NET 1.14.1 adds new methods and properties to the ElectronicValue Reader/Writer class, as well as updating some old members.

[\[+\] Expand table](#)

Updated Class	Added/Updated Members
ElectronicValueRW	<p>Methods:</p> <ul style="list-style-type: none"> • AccessData (new) • AccessLog (updated) • ActivateEVService (new) • CheckServiceRegistrationToMedium (new) • CloseDailyEVService (new) • DeactivateEVService (new) • LockTerminal (updated) • OpenDailyEVService (new) • RegisterServiceToMedium (new) • UnlockTerminal (updated) • UnregisterServiceToMedium (new) • UpdateData (new) • UpdateKey (updated) <p>Properties:</p> <ul style="list-style-type: none"> • CapMembershipCertificate (new) • CardServiceList (updated) • CurrentService (updated) • ReaderWriterServiceList (updated) • ServiceType (new) <p>Enumerations:</p> <ul style="list-style-type: none"> • AccessDataType (new) • ServiceType (new) <p>Transition events:</p> <ul style="list-style-type: none"> • TransitionNotifyProgress1To100 (new) • TransitionConfirmDeviceData (new)

Changes in the POS for .NET 1.14 Release

POS for .NET 1.14 updates the following device classes with new and updated members.

[\[+\] Expand table](#)

Updated Class	Added Members
Biometrics	<p>Added the following events:</p> <ul style="list-style-type: none"> • StatusSensorFailedRead • StatusSensorReady

Updated Class	Added Members
	<ul style="list-style-type: none"> • StatusSensorComplete
<p>ElectronicValueRW</p>	<p>Added the following methods:</p> <ul style="list-style-type: none"> • ClearParameterInformation • QueryLastSuccessfulTransactionResult • RetrieveResultInformation • SetParameterInformation <p>Added the following properties:</p> <ul style="list-style-type: none"> • CapPINDevice • CapTrainingMode • PINEntry • TrainingModeState <p>Added the following enumerations:</p> <ul style="list-style-type: none"> • PinEntryType • TrainingModeState <p>Added the following transition events:</p> <ul style="list-style-type: none"> • TransitionTouchRetry • TransitionTouchRetryCancelable • TransitionConfirmTouchRetry • TransitionConfirmCancel • TransitionNotifyInvalidOperation • TransitionConfirmInvalidOperation • TransitionConfirmRemainderSubtraction • TransitionConfirmCenterCheck • TransitionConfirmTouchTimeout • TransitionConfirmAutoCharge • TransitionNotifyCaptureCard • TransitionNotifyPin • TransitionCenterCheck • TransitionNotifyComplete • TransitionNotifyTouch • TransitionNotifyBusy • TransitionConfirmCenterCheckComplete • TransitionConfirmSelect • TransitionNotifyLock • TransitionNotifyCenterCheckComplete • TransitionConfirmPinEntryByOuterPinpad
Micr (Magnetic ink character recognition)	<p>Added support for the following country codes:</p> <ul style="list-style-type: none"> • CheckCountryCode.CMC7

Updated Class	Added Members
	<ul style="list-style-type: none"> • <code>CheckCountryCode.OTHER</code>
PosPrinter	<p>Added the following methods:</p> <ul style="list-style-type: none"> • <code>CapRecRuledLine</code> • <code>CapSlipRuledLine</code> • <code>DrawRuledLine</code> <p>Added the following enumerations:</p> <ul style="list-style-type: none"> • <code>LineDirection</code> • <code>LineStyle</code>
Scale	<p>Added the following methods:</p> <ul style="list-style-type: none"> • <code>DoPriceCalculating</code> • <code>FreezeValue</code> • <code>ReadLiveWeightWithTare</code> • <code>SetPriceCalculationMode</code> • <code>SetSpecialTare</code> • <code>SetTarePriority</code> • <code>SetUnitPriceWithWeightUnit</code> <p>Added the following properties:</p> <ul style="list-style-type: none"> • <code>CapFreezeValue</code> • <code>CapReadLiveWeightWithTare</code> • <code>CapSetPriceCalculationMode</code> • <code>CapSetUnitPriceWithWeightUnit</code> • <code>CapSpecialTare</code> • <code>CapTarePriority</code> • <code>MinimumWeight</code> • <code>ZeroValid</code> <p>Added the following enumerations:</p> <ul style="list-style-type: none"> • <code>FreezeValueType</code> • <code>PriceCalculationMode</code> • <code>SpecialTare</code> • <code>TarePriority</code>
ToneIndicator	<p>Added the following properties:</p> <ul style="list-style-type: none"> • <code>CapMelody</code> • <code>MelodyType</code> • <code>MelodyVolume</code> <p>Added the following enumeration:</p>

Updated Class	Added Members
	<ul style="list-style-type: none"> • MelodyType
BarCodeSymbology	<p>Added the following one dimensional symbologies:</p> <ul style="list-style-type: none"> • BarCodeSymbology.ItfCK • BarCodeSymbology.Gs1DataBar_Type2 • BarCodeSymbology.Ames • BarCodeSymbology.TFORMAT • BarCodeSymbology.Code39Ck • BarCodeSymbology.Code32 • BarCodeSymbology.CodeCIP • BarCodeSymbology.TriOptic39 • BarCodeSymbology.ISBT128 • BarCodeSymbology.Code11 • BarCodeSymbology.MSI • BarCodeSymbology.Plessey • BarCodeSymbology.Telepen
	<p>Added the following composite symbology:</p> <ul style="list-style-type: none"> • BarCodeSymbology.Tlc39 <p>Added the following two dimensional symbologies:</p> <ul style="list-style-type: none"> • BarCodeSymbology.Gs1DataMatrix • BarCodeSymbology.Gs1QRCode • BarCodeSymbology.Code49 • BarCodeSymbology.Code16k • BarCodeSymbology.CodablockA • BarCodeSymbology.CodablockF • BarCodeSymbology.Codablock256 • BarCodeSymbology.HANXIN
	<p>Added the following postal code symbologies:</p> <ul style="list-style-type: none"> • BarCodeSymbology.AusPost • BarCodeSymbology.CanPost • BarCodeSymbology.ChinaPost • BarCodeSymbology.DutchKix • BarCodeSymbology.InfoMail • BarCodeSymbology.JapanPost • BarCodeSymbology.KoreanPost • BarCodeSymbology.SwedenPost • BarCodeSymbology.UkPost • BarCodeSymbology.UsIntelligent • BarCodeSymbology.UsPlanet • BarCodeSymbology.PostNet

See Also

Concepts

- POS for .NET v1.14.1 SDK Documentation

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

POS for .NET v1.14.1 Features (Microsoft Point of Service for .NET)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) v1.14.1 applies the power of .NET technology to provide a flexible, easy-to-use platform that supports retail POS applications and device peripherals.

POS for .NET implements the Unified Point of Service (UnifiedPOS) v1.14.1 international standard so that retailers, hospitality organizations, and industry partner companies can quickly and reliably configure .NET-based POS applications and device peripherals for interoperability.

POS for .NET tools and classes facilitate developing POS applications and device Service Objects.

In This Section

- [POS for .NET Architecture](#) Describes the POS for .NET architecture and the relationship between POS for .NET, POS devices, and applications.
- [Service Object Overview](#) Describes how Service Objects facilitate communication between the device and the application.
- [Supported Device Classes](#) Describes the POS devices that are recognized by POS for .NET and the level of support they receive.
- [Event Management](#) Describes how event management is implemented in POS for .NET.
- [POS Exception Handling](#) Discusses the exception classes used by POS for .NET and POS for .NET error handling model.
- [POS for .NET Integration with Plug and Play](#) Describes how to integrate your POS device with Plug and Play functionality for Microsoft Windows.
- [Log Files](#) Describes the logging feature for recording POS for .NET, Service Object, and application events and debug log size limit.
- [Integration of Legacy Service Objects](#) Describes how POS for .NET supports legacy Service Objects.

Related Sections

- [Developing a POS Application](#) Outlines key POS for .NET concepts and features for developing POS applications.
- [Developing a Custom Service Object](#) Demonstrates how to create POS for .NET applications that use Service Objects to communicate with hardware devices.

See Also

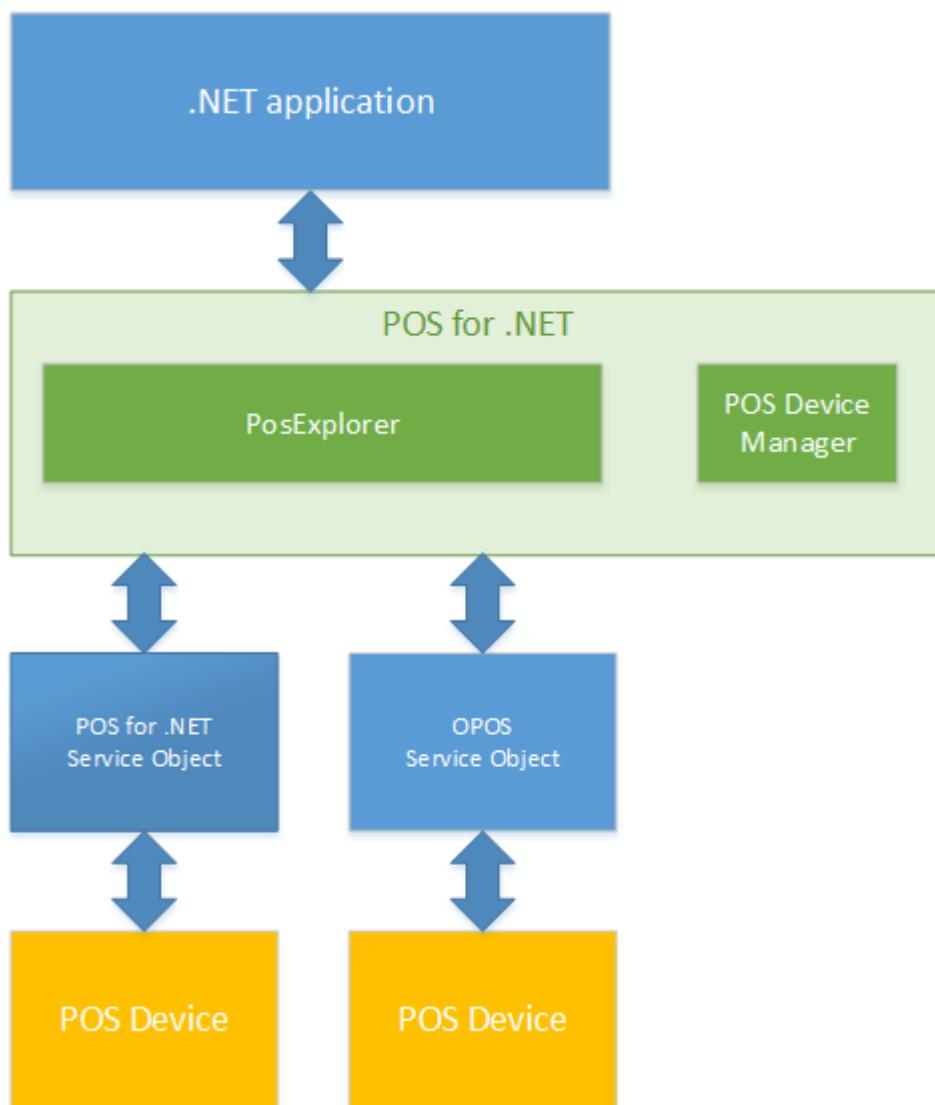
Concepts

- [What's New in POS for .NET v1.14.1](#)

POS for .NET Architecture (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The Microsoft Point of Service for .NET (POS for .NET) architecture supports both developers writing POS applications and peripheral device hardware vendors writing .NET-based Service Objects. The following illustration shows the POS for .NET architecture.



Devices use service objects to communicate with your application by using the POS for .NET interfaces.

You can use the [PosExplorer Class](#) to discover and instantiate service objects. Once you have an instance of a service object, you can use that service object to interact with the POS device.

Plug and Play devices automatically notify PosExplorer when those devices are connected or disconnected. For non-Plug and Play devices, you can use the [POS Device Manager](#) to manage how PosExplorer discovers those devices.

Support for POS Applications

The POS for .NET public API provides POS applications with information about connected POS devices. The public API also creates instances of the Service Object (SO) classes for interacting with the devices.

POS for .NET uses Plug and Play for Windows Embedded to detect POS device connection and disconnection. The public API conveys Plug and Play events to the application.

For older applications, the POS for .NET legacy interoperability system exposes legacy OLE for Retail POS (OPOS) controls as .NET Service Objects.

For more information, see [POS for .NET API Support](#).

Support for Service Objects

Hardware vendors write device-specific Service Objects that interact with peripheral hardware. The Service Objects also implement interfaces as specified in the Unified Point of Service (UnifiedPOS) v1.14 standard. With POS for .NET device classes, hardware vendors can concentrate their efforts on implementing the device-specific details.

POS for .NET offers abstract **Basic** device classes for every device type defined in the UnifiedPOS specification. The enhanced **Basic** classes, called **Base** classes, supply functionality common to all POS devices. POS for .NET provides abstract **Base** device classes for nine devices. The **Base** classes further implement core POS functionality specific to the particular device class.

POS for .NET supplies **Base** classes for the following devices:

- Cash Drawer
- Check Scanner
- Keyboard
- Line Display
- Magnetic Stripe Reader
- Pin Pad
- Printer
- Scanner

- RFID Scanner

See Also

Concepts

- [Typical POS Application Architecture](#)
- [POS for .NET Integration with Plug and Play](#)
- [Supported Device Classes](#)

Other Resources

- [POS for .NET Service Object Architecture](#)
- [POS for .NET v1.14 Features](#)
- [POS for .NET API Support](#)

Service Object Overview (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Service Objects function as the interface between an application and a POS device. Each Service Object facilitates communication between the application and its associated device by implementing one device interface. The device interfaces provide the properties, methods, and events required by unique POS devices. This enables the application to manage and read data from them.

Not all devices receive the same level of support in POS for .NET. Each POS device that POS for .NET recognizes is provided with up to three levels of **Interface** classes that provide some level of functional support. The three levels of **Interface** classes are **Interface** classes, **Basic** classes, and **Base** classes. For more information about POS for .NET **Interface** classes, see [POS for .NET Class Tree](#). For more information about the default level of support provided by Service Objects, see the individual entries for each Service Object in [Microsoft.PointOfService.BaseServiceObjects](#) and [Supported Device Classes](#).

Because each Service Object facilitates communications with a specific device, a different Service Object instance must be created for each connected peripheral device.

See Also

Reference

- [PosCommon](#)
- [Microsoft.PointOfService.BaseServiceObjects](#)

Concepts

- [POS for .NET Architecture](#)
- [Supported Device Classes](#)
- [POS for .NET Device Basic Classes](#)

Other Resources

- [Developing a Custom Service Object](#)
- [POS for .NET Service Object Architecture](#)

Supported Device Classes (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) v1.14 represents the 36 peripheral devices identified in the Unified Point of Service (UnifiedPOS) v1.14 specification by abstract **Interface** and **Basic** device classes. POS for .NET also provides nine abstract **Base** device classes that further implement core POS functionality specific to those particular device types.

Hardware vendors use the device classes to create Service Objects that link their peripheral devices to the applications.

Interface Classes

POS for .NET supplies **Interface** classes for all 36 UnifiedPOS devices. The **Interface** classes provide the entry points as specified in the UnifiedPOS specification, but offer minimal functionality.

Basic Classes

POS for .NET **Basic** classes contain basic functional support for all 36 devices. **Basic** classes provide generic support for opening, claiming, and enabling the device, device statistics, and management of delivering events to the application. In addition, each **Basic** class contains a set of inherited and protected methods that can be implemented by the Service Object.

Base Classes

For the nine primary UnifiedPOS device types, POS for .NET supplies fully functional **Base** classes that extend their corresponding **Basic** classes with device-specific members. You could think of these classes as enhanced or extended **Basic** classes. Because **Base** classes provide a nearly complete implementation, Service Object developers should derive from these classes whenever possible.

UnifiedPOS Devices and POS for .NET Device Classes

The following table lists the UnifiedPOS devices with their equivalent POS for .NET **Basic** and **Base** device classes (where applicable).

[\[+\] Expand table](#)

UnifiedPOS Device	Interface Class	Basic Class	Base Class
Belt	Belt	BeltBasic	
Biometrics	Biometrics	BiometricsBasic	
Bill Acceptor	BillAcceptor	BillAcceptorBasic	
Bill Dispenser	BillDispenser	BillDispenserBasic	
Bump Bar	BumpBar	BumpBarBasic	
Cash Changer	CashChanger	CashChangerBasic	
Cash Drawer	CashDrawer	CashDrawerBasic	CashDrawerBase
CAT - Credit Authorization Terminal	Cat	CatBasic	
Check Scanner	CheckScanner	CheckScannerBasic	CheckScannerBase
Coin Acceptor	CoinAcceptor	CoinAcceptorBasic	
Coin Dispenser	CoinDispenser	CoinDispenserBasic	
Electronic Journal	ElectronicJournal	ElectronicJournalBasic	
Electronic Value Reader / Writer	ElectronicValueRW	ElectronicValueRWBasic	
Fiscal Printer	FiscalPrinter	FiscalPrinterBasic	
Gate	Gate	GateBasic	
Hard Totals	HardTotals	HardTotalsBasic	
Image Scanner	ImageScanner	ImageScannerBasic	
Item Dispenser	ItemDispenser	ItemDispenserBasic	
Keylock	Keylock	KeylockBasic	
Lights	Lights	LightsBasic	
Line Display	LineDisplay	LineDisplayBasic	LineDisplayBase
MICR - Magnetic Ink	Micr	MicrBasic	

UnifiedPOS Device	Interface Class	Basic Class	Base Class
Character Recognition			
Motion Sensor	MotionSensor	MotionSensorBasic	
MSR - Magnetic Stripe Reader	Msr	MsrBasic	MsrBase
PIN Pad	PinPad	PinPadBasic	PinPadBase
Point Card Reader / Writer	PointCardRW	PointCardRWBasic	
POS Keyboard	PosKeyboard	PosKeyboardBasic	PosKeyboardBase
POS Power	PosPower	PosPowerBasic	
POS Printer	PosPrinter	PosPrinterBasic	PosPrinterBase
Remote Order Display	RemoteOrderDisplay	RemoteOrderDisplayBasic	
RFID Scanner	RFIDScanner	RFIDScannerBasic	RFIDScannerBase
Scale	Scale	ScaleBasic	
Scanner (Bar Code Reader)	Scanner	ScannerBasic	ScannerBase
Signature Capture	SignatureCapture	SignatureCaptureBasic	
Smart Card Reader / Writer	SmartCardRW	SmartCardRWBasic	
Tone Indicator	ToneIndicator	ToneIndicatorBasic	

See Also

Reference

- [Microsoft.PointOfService](#)
- [Microsoft.PointOfService.BaseServiceObjects](#)

Concepts

- [What's New in POS for .NET v1.14](#)

Other Resources

- [POS for .NET v1.14 Features](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

Event Management (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Events management represents one of the key aspects of programming applications for Microsoft Point of Service for .NET (POS for .NET). All input in the POS for .NET system is event-driven, and each segment of the [POS for .NET Architecture](#) uses events to communicate with the other applications and Service Objects.

Event-Driven Processing Model

Event-driven input begins when an attached POS device receives data input. If that device is enabled (the **DeviceEnabled** property is set to **true**), then received data will be queued as a **DataEvent** event and sent to the application. Events are delivered in a first-in, first-out manner by an internal service thread. Just before this event is raised, a Service Object may use the **PreFireEvent** method to update properties before that event is sent off.

After the event data is received, the device will automatically disable itself (setting the **DeviceEnabled** property to **false**) if the **AutoDisable** property is set to **true**. While disabled, the device cannot queue new input, and the physical device will be disabled, if possible.

When the application is ready to receive input from the device, it sets the **DataEventEnabled** property to **true**. The application then starts to receive queued **DataEvent** events, even if those **DataEvent** events were queued before **DataEventEnabled** property was set to **true**.

Additional data events may be disabled by setting the **DataEventEnabled** property or the **FreezeEvents** property to **false**. This causes later input data to be queued while the application processes the current input and associated properties. When the application is ready for more data, it may re-enable events by setting the **DataEventEnabled** property to **true**.

Event-Driven Input and Device Sharing

If the input device is an exclusive-use device, the application must both claim and enable the device before it uses it to read input.

If the device is shareable, one or more applications must open and enable the device before it uses it to read input. An application must call the **Claim** method to request exclusive access to the device before the Service Object sends data to it by using **DataEvent**. If event-driven input is received but the device remains unclaimed, the input is buffered until an application claims the device and the **DataEventEnabled** property is set to **true**. This behavior promotes orderly sharing of the device between multiple applications, effectively passing the input focus between them.

Event-Driven Input and Error Handling

The device enters an error state if an error is encountered while receiving event-driven input. Then it queues an **ErrorEvent** event (that contains **InputData** or **InputErrorEvent** loci). These events are not delivered until the **DataEventEnabled** property is set to **true** to guarantee orderly application sequencing. Each **ErrorEvent** indicates which one of two possible error loci is responsible:

- **InputData** –Used if the error occurred while one or more **DataEvent** events are queued. The **ErrorEvent** jumps to the head of the event queue for immediate handling so that the application can immediately respond by clearing input or notifying the user of the error. Then finish processing the buffered input.
- **Input** – Used if an error has occurred and no data is available. If input data is already queued when the error occurs, an **ErrorEvent** with the **InputData** locus is queued and delivered first, then the remaining **DataEvents** in the queue are raised and handled. Finally, an **ErrorEvent** with the **Input** value is sent to indicate that the queue is empty and no data is available. It is significant to notice that if an **ErrorEvent** with the **InputData** value was delivered and the application event handler responded with a **Clear** value, this **InputDataErrorEvent** is not delivered. Typically, this error is entered at the end of the event queue.

The device may exit the Error state when one of the following occurs:

- The application returns from the **InputErrorEvent**. The application returns from the **InputDataErrorEvent** with a **Clear** value for the **ErrorResponse** property.
- The application calls the **ClearInput** method.

For some devices, the application must call a method to begin event-driven input. After the input is received by the Service Object, then typically no additional input will be received until the method is called again. Examples for devices that use this variation of event-driven input, also known as asynchronous input, include the magnetic ink character recognition (MICR) and Signature Capture devices. The **DataCount** property can be read to obtain the number of **DataEvent** events in the queue.

All input in the queue can be deleted by calling the **ClearInput** method. **ClearInput** may be called after **Claim** for exclusive-use devices or **Open** for shareable devices.

The general event-driven input model does not prevent the definition of device classes that contain methods or properties that return input data directly. An example of this variation of event-driven input, also known as synchronous input, is the **Keylock** device.

Event Types

POS for .NET implements Unified Point Of Service (UnifiedPOS) events as standard .NET events with multicast delegates. The events inform an application of various activities or changes with a device, such as when a device is added or removed. The following table lists the event types.

Event	Description
DataEvent	An event raised by the Service Object to notify the application that input data is available.
ErrorEvent	An event raised by the Service Object to notify the application that a device error has occurred and that a suitable response by the application is necessary to process the error condition.
StatusUpdateEvent	An event raised by the Service Object to alert the application of a device status change.
OutputCompleteEvent	An event raised by the Service Object to notify the application that the queued output request has been completed successfully.
DirectIOEvent	An event raised by the Service Object to communicate information directly to the application.

The Service Object must stack these events on an internally created and managed queue. Events are delivered in a first-in, first-out manner and are delivered by an internal service thread.

The following conditions cause event delivery to be delayed until the condition is corrected:

- The application has set the **FreezeEvents** property to **true**. The **FreezeEvents** property allows events to be queued but prevents their delivery until **FreezeEvents** is set to **false**.
- The event is a **DataEvent** or an input **ErrorEvent** but the property **DataEventEnabled** is **false**.

Rules for event queue management are as follows:

- The device may only queue new events while the device is enabled.
- The device delivers queued events until the application calls the **Close** method or, for exclusive-use devices, the **Release** method. When these methods are called, any remaining events in the queue are deleted.
- The **ClearInput** method clears **DataEvents** and input **DeviceErrorEvents** (**ErrorLocus** = **Input** or **InputData**).
- The **ClearOutput** method clears output **DeviceErrorEvents** (**ErrorLocus** = **Output**).

See Also

Reference

- [FreezeEvents](#)
- [StatusUpdateEvent](#)
- [DirectIOEvent](#)
- [ErrorLocus](#)

Concepts

- [Device Output Models](#)
- [POS Exception Handling](#)

POS Exception Handling (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Error handling in Microsoft Point of Service for .NET (POS for .NET) is built on the object-oriented model of throwing and catching exceptions. Different exceptions are thrown in response to run-time errors, and each exception contains information about the error that triggered it in the form of an [ErrorCode](#).

The **ErrorCode** property of a thrown exception holds information about the cause of the error. Possible values for this property represent the full set of standard Unified Point Of Service (UnifiedPOS) error codes. For more information about the mapping between UnifiedPOS error codes and the POS for .NET **ErrorCode** values, see [POS for .NET Exception Classes](#).

POS for .NET provides four exception classes to help applications better handle errors. These are [PosException](#), [PosControlException](#), [PosManagementException](#), and [PosLibraryException](#):

- **PosException** is an abstract class that holds general exception data. **PosException** fulfills a role similar to that of the **System.Exception** class in the .NET Framework, and is the POS for .NET implementation of the **UposException** class in the UnifiedPOS specification. All other POS for .NET exception classes are derived from **PosException**.
- **PosControlException** is the standard exception thrown by POS for .NET Service Objects. **PosControlException** contains a [ErrorCode](#) property which holds information about the cause of the exception.
- **PosLibraryException** holds the exception data generated by **PosExplorer** during class operations. **PosLibraryException** does not contain an [ErrorCode](#) property.
- **PosManagementException** holds the exception data generated by POS for .NET management APIs. **PosManagementException** does not contain an [ErrorCode](#) property.

Error Handling in POS for .NET

Error handling in POS for .NET is compliant with UnifiedPOS specification guidelines. Error handling is event-driven, uses error codes to store exception information, and is largely implementation-specific.

Handling errors in POS for .NET follows this general procedure:

1. An error is thrown by event-driven input.
2. The device changes its [State](#) property to indicate that it has encountered an error.
3. An **ErrorEvent** event is queued to alert the application that an error has occurred. The **ErrorEvent** is added to the end of the queue.
4. If one or more **DataEvent** events are queued in front of the **ErrorEvent** event, another **ErrorEvent** is queued and added at the head of queue. This warns the application about the error quickly so it can respond in an implementation-specific manner prior to processing any queued **DataEvents**.
5. If the applications properties are configured to accept events (**DataEventEnabled** is **true** and **FreezeEvents** is **false**), it responds to the **ErrorEvent** event in a manner determined by the [ErrorResponse](#) property, as indicated in the following table.

Value	Meaning of Response
Clear	Clears any buffered DataEvent events and ErrorEvent events, exits the Error state, and changes the device State to Idle.
ContinueInput	Acknowledges the error and directs the device to continue processing. The device remains in the Error state and will deliver additional data events as directed by the DataEventEnabled property. When all input has been delivered, and the DataEventEnabled property is again set to true, another ErrorEvent is delivered with the locus Input.
Retry	Directs the device to retry the input. The Error state is exited, and State is changed to Idle. This response is only selected when the device chapter specifically allows it and when the locus is Input.

The application may also take implementation-specific steps to respond to the error at this time.

See Also

Reference

- [PosException](#)
- [PosLibraryException](#)
- [PosManagementException](#)
- [PosControlException](#)
- [ErrorCode](#)
- [PosExplorer](#)

Concepts

- [PosExplorer Class](#)
- [Exception Classes](#)
- [Event Management](#)

POS for .NET Integration with Plug and Play (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) leverages Windows Embedded Plug and Play technology specifications to detect peripheral POS devices that are enabled for Plug and Play. Plug and Play support simplifies the installation and maintenance of POS devices.

PosExplorer

The POS for .NET [PosExplorer](#) class serves as the interface between Plug and Play notifications and POS applications. **PosExplorer** translates the relevant Plug and Play notifications into POS for .NET events, which it then sends to the POS application.

Plug and Play Events

The **PosExplorer** class exposes two Plug and Play events for use by POS applications:

- [DeviceAddedEvent](#) The **DeviceAddedEvent** triggers when a POS device is connected to the system.
- [DeviceRemovedEvent](#) The **DeviceRemovedEvent** triggers when a POS device is disconnected from the system.

See Also

Reference

- [DeviceAddedEvent](#)
- [DeviceRemovedEvent](#)

Concepts

- [PosExplorer Class](#)
- [Supported Device Classes](#)
- [Plug and Play Support](#)

Other Resources

- [POS for .NET v1.14 Features](#)

Log Files (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) includes a logging feature for recording POS for .NET, Service Objects, and application events. Logging parameters are read from the POS for .NET registry key, `\HKLM\SOFTWARE\POSfor.NET\Logging`, and entries are written using the [Logger](#) class.

Enabling Logging

Logging is enabled when the registry key `\HKLM\SOFTWARE\POSfor.NET\Logging\Enabled` is set to any non-zero value.

Log File Size

The maximum log file size is specified in the registry key `\HKLM\SOFTWARE\POSfor.NET\Logging\MaxLogFileSizeMB`. If this file size is exceeded while logging is enabled, logging will stop. There will be no exception or error returned to the application.

By default, the maximum log file size is 10 megabyte (MB).

Log File Location

The registry key `\HKLM\SOFTWARE\POSfor.NET\Logging\location` is used to determine where log files will be written.

By default, this location is set to the environment variable `%TEMP%` which, in Windows, defaults to the directory `C:\Documents and Settings\username\Local Settings\temp`. This is a per-user directory.

Log File Names

Log file names are composed of three elements:

- The base file name contained in the registry key `\HKLM\SOFTWARE\POSfor.NET\Logging\Name`. The default for this value is `PosFor.Net`.

- A timestamp in this format: (yyyy-mm-dd hh-mm-ssZ)
- The file extension .txt.

This is an example of a typical log file name:

PosFor.Net(2006-08-10 18-33-29Z).txt

Log File Header

A header containing information such as the user, OS, calling thread, and process is written to each log file when it is created. This header includes the following fields:

- **Current user:** The name of the current user.
- **Computer name:** The name of the computer creating the log.
- **OS version:** The version of Windows that is being run, including service packs.
- **.Net runtime:** The version of the .NET runtime.
- **Process Id:** The PID of the process that created the log file.
- **Thread Id:** The thread that created the log.
- **Max log file size:** The maximum file size to be used for this log file.
- **File:** The name of the executable that created the log file.
- **InternalName:** The internal name of the executable.
- **OriginalFilename:** The original name of the executable.
- **FileVersion:** The version information stored in the executable.
- **FileDescription:** The description stored in the executable.
- **Product:** The product description stored in the executable.
- **ProductVersion:** The file version stored in the executable.
- **Debug:** Debug flag.
- **Patched:** Patched file.
- **PreRelease:** Pre-release flag.
- **PrivateBuild:** Private build flag.
- **SpecialBuild:** Special build flag.
- **Language:** The language used to create the log file.

Log File Entries

Log entries can be created by POS for .NET or by either the application or the Service Object. Entries are created by calling the appropriate method on an instance of the **Logger** class.

Each entry contains the following fields:

- Timestamp.

- Thread ID that created the entry.
- Importance level. Each log entry is marked with its level of importance which is determined by which **Logger** method is involved.

Importance Tag in Log Entries	Corresponding Logger Method
INFO	Logger.Info
WARNING	Logger.Warning
ERROR	Logger.Error

- Name string specified by the code that called the **Logger** method. This string is specified when the **Logger** method is invoked and may not necessarily contain the name of the executable.
- For example, a typical entry in the log file would look like this: [8/10/2006 6:12:14 PM 2936 INFO PosExplorer] Entering LoadExplorer()

Comments

If there is a log file open, and the application calls the [Refresh\(\)](#) method, the file is closed and a new one created with the updated time stamp.

See Also

Reference

- [Logger](#)

Concepts

- [POS for .NET Registry Settings](#)
- [Plug and Play Support](#)

Other Resources

- [System Configuration](#)

Integration of Legacy Service Objects (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) supports both .NET-based Service Objects and legacy OLE for Retail POS (OPOS), COM-based Service Objects. POS for .NET provides applications with the same interface to both generations of Service Objects.

See Also

Concepts

- [POS for .NET FAQ](#)
- [What's New in POS for .NET v1.14](#)

Other Resources

- [POS for .NET v1.14 Features](#)

Developing a POS Application (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) provides classes and tools that make writing robust POS applications simple and easy.

Service Object classes represent POS peripheral devices, supplying the properties, methods, and events that are defined in the Unified Point Of Service (UnifiedPOS) Retail Peripheral Architecture specification.

The [PosExplorer](#) class lets applications enumerate installed POS devices, instantiate Service Objects for them, and receive Plug and Play events when a POS peripheral device is connected or disconnected.

In This Section

- [POS for .NET Application Compatibility with 32-bit OPOS Service Objects](#) Describes how to use 32-bit OLE for Retail POS (OPOS) service objects with a 64-bit OS.
- [Typical POS Application Architecture](#) Describes the POS for .NET application architecture and organization.
- [POS for .NET API Support](#) Provides an overview of POS for .NET API support and links to specific important API classes and conceptual topics.
- [Event Handler Sample](#) Demonstrates the use of POS for .NET events in an application.

Related Sections

- [POS for .NET v1.14 Features](#) Provides a high-level overview of the POS for .NET system.
- [Developing a Custom Service Object](#) Demonstrates how to create a POS for .NET application, which uses Service Objects to communicate with hardware devices.

POS for .NET Application Compatibility with 32-bit OPOS Service Objects (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Because OLE for Retail POS (OPOS) service objects only exist as 32-bit objects, under certain circumstances 64-bit applications that reference Microsoft Point of Service for .NET (POS for .NET) assemblies may fail to load OPOS service objects. If your application meets certain requirements, you can use one of the two procedures discussed in this topic to interoperate with 32-bit OPOS service objects.

POS for .NET applications that are compiled as 32-bit should work with all OPOS service objects, and do not require any of the procedures described in this topic.

Requirements

Your application must meet all of the following conditions in order to use the procedures described in this topic:

- Your application runs on 64-bit Windows.
- Your application uses OPOS service objects.
- The executable that references POS for .NET is compiled as Microsoft intermediate language (MSIL) ('anycpu' platform).
- All managed assemblies referenced by the application are also compiled as MSIL.

Under these conditions, the Common Library Runtime (CLR) will see that all managed assemblies are MSIL so it will choose to run the managed application as a 64-bit process. When POS for .NET attempts to load a 32-bit OPOS service object as an in-process COM server, it will fail to load and the device will not be visible to your application. This is because a 64-bit process cannot load a 32-bit COM server into its process space.

You can work around this by using one of the following two solutions:

Compile the managed process as a 32-bit process

You can force your process to run as a 32-bit process by compiling your main executable to target the **x86** or **anycpu32bitpreferred** platform. This causes the managed app to run as 32-bit and load the OPOS object as an in-process COM server.

To compile your application as 32-bit at the command prompt

1. Add the `/platform:x86` compiler option to your C# compiler command, as in the following example:

```
csc /platform:x86 <filename>.cs
```

For more information, see [/platform \(C# Compiler Options\)](#) on MSDN.

To compile your application as 32-bit in Visual Studio 2013

1. In Microsoft Visual Studio 2013, open your project.
2. Open the **BUILD** menu and select **Configuration Manager**.
3. In the **Configuration Manager** dialog box, in the **Platform** column, click on the cell to expand the drop down menu and select **x86**. If **x86** is not available, select **<New...>**, and then select **x86** as the new platform and click **OK**.
4. Rebuild the project.

Compile the managed process as a 64-bit process and modify the COM registry of the OPOS object

You can modify the COM registration of the OPOS service object to use a 32-bit host process. This causes Windows to handle inter-process communication (IPC) and marshalling of data between the 64-bit managed process and the 32-bit COM surrogate host process.

To modify the COM registry of the OPOS service object

1. In the registry, locate your COM object GUID key under **HKEY_CLASSES_ROOT/Wow6432Node/CLSID**.
2. Once you locate the COM object GUID key, add a new string value (REG_SZ). Set the name to **AppID** and set the data to the COM object GUID, including the curly braces.
3. Add a new key under **HKEY_CLASSES_ROOT/Wow6432Node/AppID** with the same name as the COM object GUID key.
4. Under the new key you just added, add a new string value (REG_SZ). Set the name to **DllSurrogate**. Leave the value empty.
5. Create a new key under **HKEY_LOCAL_MACHINE/Software/Classes/AppID** with the same name as the COM object's GUID, if it doesn't already exist. You do not need to add any values to this key.

Alternatively, you can use the following Windows PowerShell script to modify the COM registry for all of the OPOS Common Control Objects (CCO) to use out of process COM servers. You can run this script to ensure that all OPOS service objects will be able to interoperate with 64-bit applications. You must run the script from an administrator Windows PowerShell prompt.

C#

```
# This Windows PowerShell script modifies the COM registry for all OPOS
# Common Control Objects (CCO) so that they use out of process COM servers.
# This enables OPOS service objects to work with both 32-bit and 64-bit
# POS for .NET applications.

# .Synopsis
# Create-Regkey: This function creates a new key in the registry
function Create-Regkey {
    param(
        [string] $Key
    )

    if (!(test-path -path $Key -pathType container)) {
        New-Item -path $Key -type container | Out-Null
    }
}

# .Synopsis
# Set-RegEntry: This function creates a new registry key in the registry and
# creates a new value in the key.
function Set-RegEntry {
    param(
        [string] $Key,
        [string] $Name,
```

```

        [string] $.PropertyType,
        $Value
    )

Create-RegKey -Key $Key
Remove-ItemProperty -Path $Key -Name $Name -ErrorAction SilentlyContinue
New-ItemProperty -Path $Key -Name $Name -PropertyName $Name -PropertyType $.PropertyType -
Value $Value | Out-Null
}

# Iterate through all of the OPOS Common Control Objects, setting registry
# entries and values for each object.

for ($i = 2; $i -lt 38; $i++) {
    $clsid = '{CCB90{0:D2}2-B81E-11D2-AB74-0040054C3719}' -f $i

    Set-RegEntry -Key "hklm:\SOFTWARE\Classes\Wow6432Node\CLSID\$clsid" -
    Name 'AppID' -PropertyType String -Value $clsid
    Set-RegEntry -Key "hklm:\SOFTWARE\Classes\Wow6432Node\AppID\$clsid" -
    Name 'DllSurrogate' -PropertyType String
    Create-RegKey -Key "hklm:\SOFTWARE\Classes\AppID\$clsid"
}

```

If you need to revert the COM registry after running the previous script, you can run the following Windows PowerShell script to [remove](#) the [new](#) COM registry entries:

```

# This Windows PowerShell script restores the COM registry for all OPOS
# Common Control Objects (CCO) to their original values.

for ($i = 2; $i -lt 38; $i++) {
    $clsid = '{CCB90{0:D2}2-B81E-11D2-AB74-0040054C3719}' -f $i

    Remove-ItemProperty -Path
    "hklm:\SOFTWARE\Classes\Wow6432Node\CLSID\$clsid" -Name 'AppID'
    Remove-Item -Path "hklm:\SOFTWARE\Classes\Wow6432Node\AppID\$clsid"
    Remove-Item -Path "hklm:\SOFTWARE\Classes\AppID\$clsid"
}

```

See Also

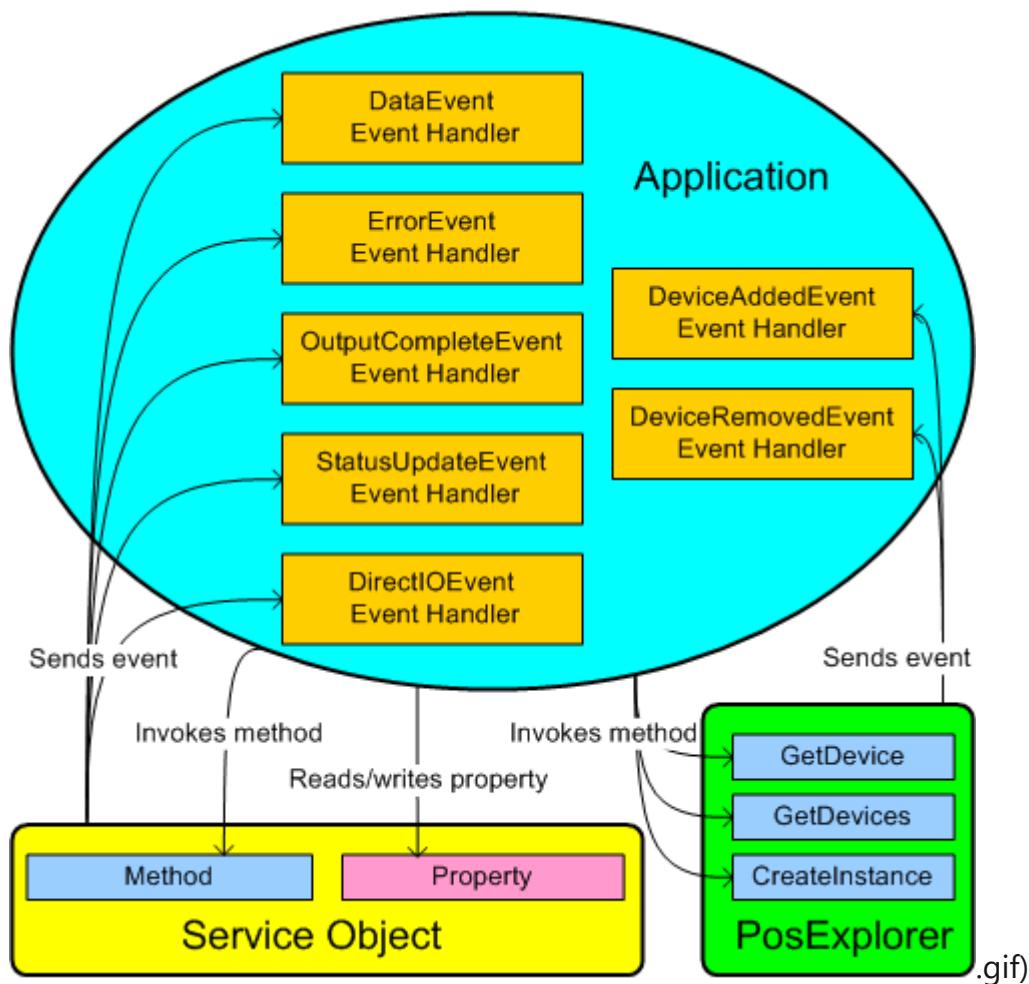
Other Resources

- [Developing a POS Application](#)

Typical POS Application Architecture (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The principal elements of POS application architecture are event handlers and the public interface to Service Objects and the Microsoft Point of Service for .NET (POS for .NET) [PosExplorer Class](#), as shown in the following illustration.



Event Handlers

POS for .NET uses five Service Object events to inform POS applications of POS device activities or changes, as specified in the Unified Point Of Service (UnifiedPOS) standard.

The [PosExplorer Class](#) translates Plug and Play notifications into events that it sends to applications.

Applications implement event handlers to manage these events.

Public Interface to Service Objects

Service Objects expose public methods and properties to POS applications, including those defined in the UnifiedPOS standard.

POS applications use those methods and properties to get device status and to transfer data to and from devices.

Public Interface to PosExplorer

The **PosExplorer** class exposes public methods to POS applications that create Service Objects and provide information about Service Objects and devices.

See Also

Concepts

- [POS for .NET Architecture](#)
- [PosExplorer Class](#)

Other Resources

- [POS for .NET Service Object Architecture](#)
- [Developing a POS Application](#)

POS for .NET API Support (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The Microsoft Point of Service for .NET (POS for .NET) class library supports the development of POS applications with **Base** classes that support POS devices, exceptions, and events. The POS for .NET class library also provides applications with the ability to discover and load Service Objects using the [PosExplorer](#) and [DeviceInfo](#) classes.

In This Section

- [PosExplorer Class](#) Describes the **PosExplorer** class, a key component of POS for .NET functionality.
- [DeviceInfo Class](#) Describes the **DeviceInfo** class and implementation.
- [Exception Classes](#) Describes the POS exception classes and their associated **ErrorCode** properties.
- [PosCommon Class](#) Describes the **PosCommon** class, the **Base** class upon which each Service Object class or interface is built.

Related Sections

- [Typical POS Application Architecture](#) Illustrates the architecture model of a POS application.
- [Developing a POS Application](#) Provides an overview of POS for .NET API support and supplies links to important POS for .NET device classes and conceptual Help topics.
- [Microsoft.PointOfService](#) Contains the POS for .NET API topics, including the **PosExplorer** and **PosCommon** classes, as well as interfaces for specific POS devices.

PosCommon Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

PosCommon is the **Base** class for all specific **Interface** classes, and all Service Objects indirectly derive from it. **PosCommon** defines the common properties, methods, and events that the Unified Point Of Service (UnifiedPOS) specification requires in all device classes.

PosCommon Properties

The following table describes the properties of the **PosCommon** class available to POS applications.

[\[+\] Expand table](#)

Property	Type	Description
CapCompareFirmwareVersion	bool	Indicates whether the Service Object and device supports comparing the firmware version in the physical device against that of a firmware file.
CapPowerReporting	PowerReporting enum	Indicates the power reporting capabilities of the device.
CapStatisticsReporting	bool	Indicates whether the device can accumulate and can provide various statistics regarding usage.
CapUpdateStatistics	bool	If set to true, some or all of the device statistics can be reset to 0 (zero) using the ResetStatistic method for one update and ResetStatistics method for a list of updates, or updated using the UpdateStatistic method for one update and the UpdateStatistics method for a list of updates with the corresponding specified values.
CapUpdateFirmware	bool	Indicates whether the device's firmware can be updated via the UpdateFirmware method.

Property	Type	Description
CheckHealthText	string	Indicates the health of the device.
Claimed	bool	Indicates whether the device is claimed for exclusive access.
DeviceDescription	string	Holds a string identifying the device and the company that manufactured it.
DeviceEnabled	bool	Indicates whether the device is in an operational state.
DeviceName	string	UnifiedPOS calls it PhysicalDeviceName; OLE for Retail POS (OPOS) calls it DeviceName.
DevicePath	string	Set by POS for .NET for Plug and Play devices. For non-Plug and Play devices, DevicePath can be assigned using a configuration file.
FreezeEvents	bool	When set to true, the application has requested that the Service Object not deliver events.
PowerNotify	PowerNotification enum	Holds the type of power notification selection made by the application.
PowerState	PowerState enum	Holds the current power condition.
ServiceObjectDescription	string	Identifies the Service Object supporting the device and the company that produced it. This property is listed as DeviceServiceDescription in the UnifiedPOS specification.
ServiceObjectVersion	System.Version	Holds the Service Object version number. This property is listed as DeviceServiceVersion in the UnifiedPOS specification.
State	ControlState enum	Holds the current state of the device.
SynchronizingObject	ISynchronizeInvoke	Gets or sets the marshalling object for event handler calls from a POS event.

PosCommon Methods

The following table describes the methods of the **PosCommon** class available to applications.

[Expand table](#)

Method	Return Type	Description
CheckHealth	string	Performs a health check on the device. The type of check to be performed is indicated by the HealthCheckLevel parameter. The method also updates the CheckHealthText property.
Claim	void	Requests exclusive access to the device. Service Object writers are advised to only throw exceptions in unexpected conditions; for example, OutOfMemory . Otherwise, Service Objects should return True if the device was claimed and False if a time-out occurred.
Close	void	Releases the device and its resources.
CompareFirmwareVersion	CompareFirmwareResult	Determines whether the version of the specified firmware is newer than, older than, or the same as the version of firmware in the physical device.
DirectIO	DirectIOData	Used to communicate directly with the Service Object. In the UnifiedPOS specification, it has two in/out parameters. As used by POS for .NET, this method returns a structure and no in/out parameters.
Open	void	Opens a device for subsequent input/output processing.
Release	void	Releases exclusive access to the device.
ResetStatistic	void	Resets the specified statistic to zero. Used in POS for .NET for operations on a single statistic.
ResetStatistics	void	Resets all statistics for a specified category to 0 (zero).
ResetStatistics	void	Resets the specified statistics to 0 (zero).
ResetStatistics	void	Resets all statistics associated with a

Method	Return Type	Description
		device to 0 (zero).
RetrieveStatistic	string	Retrieves the specified device statistic. Used in POS for .NET for operations on a single statistic.
RetrieveStatistics	string	Retrieves all device statistics.
RetrieveStatistics	void	Retrieves the statistics for the specified category.
RetrieveStatistics	void	Retrieves the specified statistics.
UpdateFirmware	void	Updates the firmware of a device with the version of the firmware contained in the specified filename.
UpdateStatistic	void	Updates a statistic. Added to POS for .NET for operations on a single statistic.
UpdateStatistics	void	Updates a list of statistics with the corresponding specified values.
UpdateStatistics	void	Updates the specified category of statistics with the specified value.

PosCommon Events

The following table describes the **PosCommon** class events.

 [Expand table](#)

Method	Description
DirectIOEvent	Raised by the Service Object to communicate information directly to the application.
StatusUpdateEvent	Raised by the Service Object to alert the application of a device status change.

Example

The following code example demonstrates how to use the properties and methods common to all Service Objects to display information about a connected device.

C#

```
// Create a derived class of PosCommon
public class PosCommonSample: PosCommon
{
    // Implement all base methods and properties.
    // ...
}

// Create instances for the example.
PosExplorer explorer = new PosExplorer();
PosCommonSample pcs = new PosCommonSample();
DeviceInfo device = explorer.GetDevice("MSR");
pcs = (PosCommonSample)explorer.CreateInstance(device);

// Open and claim the device, then print information
// about the device to the console.
pcs.Open();
pcs.Claim(1000);
Console.WriteLine("Name: {0}", pcs.DeviceName);
Console.WriteLine("Description: {0}", pcs.DeviceDescription);
Console.WriteLine("Path: {0}", pcs.DevicePath);
Console.WriteLine("Enabled: {0}", pcs.DeviceEnabled);

pcs.Close();
```

See Also

Concepts

- POS for .NET Class Tree

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

PosExplorer Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

[PosExplorer](#) provides Point of Service (POS) applications with a single entry point to Microsoft Point of Service for .NET (POS for .NET) services. **PosExplorer** supports applications by:

- Enumerating installed POS devices.
- Instantiating Service Objects.
- Receiving Plug and Play events when a POS peripheral device is connected or disconnected.

PosExplorer Properties

The following table describes **PosExplorer** properties.

Property	Type	Description
PosRegistryKey	string	Returns POS for .NET configuration root registry key relative to HKEY_LOCAL_MACHINE.
StatisticsFile	string	Returns a path to the file where device statistics are contained.
SynchronizingObject	ISynchronizeInvoke	Holds the ISynchronizeInvoke object.

PosExplorer Methods

The following table describes **PosExplorer** methods.

Method	Return Type	Description
CreateInstance	PosDevice	Instantiates a Service Object for the device.
GetDevice	DeviceInfo	Returns a device of the specified type (must be only one in the system).
GetDevice	DeviceInfo	Returns a device of the type with the specified logical name or alias.
GetDevices	DeviceCollection	Returns all POS devices.

Method	Return Type	Description
GetDevices	DeviceCollection	Returns all POS devices with the specified compatibility level.
GetDevices	DeviceCollection	Returns POS devices of the type.
GetDevices	DeviceCollection	Returns POS devices of the type and compatibility level.
Refresh	None	Re-enumerates the list of attached POS devices and rebuilds internal data structures.

PosExplorer Events

The following table describes **PosExplorer** events.

Event	Description
DeviceAddedEvent	Received when a Plug and Play-compatible POS device is connected.
DeviceRemovedEvent	Received when a Plug and Play-compatible POS device is disconnected.

Example

The following code example demonstrates how to create an instance of **PosExplorer**, connect to Plug and Play events, and use it to identify all connected Magnetic Stripe Reader (MSR) devices. The code example prints information about the MSR to the console and closes the device after it has finished.

C#

```
// Creates a new instance of an MSR.
void CreateMsr(DeviceInfo msrinfo)
{
    msr = (Msr)explorer.CreateInstance(msrinfo);
    msr.Open();
    msr.Claim(1000);
    msr.DeviceEnabled = true;
}

static void Main(string[] args)
{

    // Create a new instance of PosExplorer and use it to
    // collect device information.
    PosExplorer explorer = new PosExplorer();
    DeviceCollection devices = explorer.GetDevices();

    // Search all connected devices for an MSR, print its service
}
```

```
// object name to the console, and close it when finished.
foreach (DeviceInfo device in devices)
{
    if (device.Type == DeviceType.Msr)
    {
        if (device.ServiceObjectName == currentMsr)
        {
            CreateMsr(device);
            Console.WriteLine(device.ServiceObjectName);

            // It is important that applications close all open
            // Service Objects before terminating.
            msr.Close();
            msr = null;
        }
    }
}
```

See Also

Concepts

- [POS for .NET Integration with Plug and Play](#)

Other Resources

- [POS for .NET API Support](#)
- [Developing Service Objects Using Base Classes](#)

DeviceInfo Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The [DeviceInfo](#) class supplies Microsoft Point of Service for .NET (POS for .NET) applications with information about POS devices and the Service Objects associated with them. The [PosExplorer](#) methods, `GetDevice(String, String)` and `GetDevices()` return instances of `DeviceInfo`.

DeviceInfo Properties

The following table shows the `DeviceInfo` properties.

Property	Type	Description
Compatibility	DeviceCompatibilities	Lists the valid compatibility levels for a POS device (an OLE for Retail POS (OPOS) or .NET Service Object).
Description	string	Describes the Service Object.
HardwareDescription	string	Describes the physical device.
HardwareId	string	Provides the ID of the physical device.
HardwarePath	string	Provides the physical hardware path of the device.
IsDefault	bool	Returns true if the device is the default for its type.
LogicalNames	strings[]	Provides the alternative name(s) assigned to the device in the global configuration file by POS Device Manager (POSDM).
ManufacturerName	string	Provides the physical device manufacturer name.
ServiceObjectName	string	Provides the name of the Service Object.
ServiceObjectVersion	Version	Provides the Service Object version.
DeviceType	string	Provides the physical device type.
UposVersion	Version	Provides the UPOS version number.

DeviceInfo Methods

The following table shows the `DeviceInfo` methods.

Method	Return Type	Description
<code>IsDeviceInfoOf</code>	<code>bool</code>	Returns true if the Service Object corresponds to the <code>DeviceInfo</code> class properties.
<code>ToString</code>	<code>string</code>	Returns a string that describes the properties of the device.

See Also

Concepts

- [PosExplorer Class](#)
- [Exception Classes](#)

Other Resources

- [Developing a POS Application](#)
- [POS for .NET API Support](#)
- [POS Device Manager](#)

Exception Classes (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) error handling is implemented through the use of *exceptions*. The four POS for .NET exception classes are as follows:

- [PosException](#)
- [PosControlException](#)
- [PosManagementException](#)
- [PosLibraryException](#)

Standard Unified Point Of Service (UnifiedPOS) error codes are represented by the [ErrorCode](#) enumeration.

PosException

PosException is the **Base** exception class for **PosControlException**, **PosManagementException**, and **PosLibraryException**. **PosException** is derived from **System.Exception**.

PosControlException

PosControlException is the standard exception thrown by Service Objects to POS for .NET applications.

PosManagementException

PosManagementException is thrown by POS for .NET Device Management. Applications and Service Objects must not throw **PosManagementException**.

PosLibraryException

PosLibraryException is thrown by [PosExplorer](#). Applications and Service Objects must not throw **PosLibraryException**.

Error Codes

The following table provides a mapping between the UnifiedPOS standard error codes and the **ErrorCode** values that POS for .NET provides.

POS ErrorCode Member	UnifiedPOS Error Code	Error Cause
Busy	E_BUSY	The current Service Object state does not allow this request.
Claimed	E_CLAIMED	Another Service Object instance has already claimed the POS device.
Closed	E_CLOSED	The POS device is closed.
Deprecated	E_DEPRECATED	The method has been deprecated and is no longer available.
Disabled	E_DISABLED	The operation cannot be performed while the device is disabled.
Exists	E_EXISTS	The file name or other specified value already exists.
Extended	E_EXTENDED	A device-specific error condition occurred.
Failure	E_FAILURE	The POS device cannot perform the requested procedure, even though the device is connected to the system and active.
Illegal	E_ILLEGAL	The POS application attempted an illegal or unsupported operation with the device, or used an invalid parameter value.
NoExist	E_NOEXIST	The file name or other specified value does not exist.
NoHardware	E_NOHARDWARE	The POS device is not connected to the system or is not turned on.
NoService	E_NOSERVICE	The Service Object cannot communicate with the device, normally because of a setup or configuration error.
NotClaimed	E_NOTCLAIMED	The POS application attempted to access an exclusive-use device that must be claimed before the method or property set action can be used.
Offline	E_OFFLINE	The POS device is offline.
Timeout	E_TIMEOUT	The Service Object timed out waiting for a response from the POS device.

Example

The following code example demonstrates how MSR handles POS exceptions and uses the **ErrorCode**s contained in those exceptions to gather information about them.

C#

```
// Create a new instance of the MSR and opens the device.  
msr = (Msr)explorer.CreateInstance(msrinfo);  
msr.Open();  
  
// Try to enable the device without first claiming it.  
// This will throw a PosControlException which, through  
// its ErrorCode, will yield information about the exception.  
try  
{  
    msr.DeviceEnabled = true;  
}  
catch (PosControlException e)  
{  
  
    // Examine the ErrorCode to determine the cause of the error.  
    if (e.ErrorCode == ErrorCode.NoHardware)  
    {  
        Console.WriteLine("The POS device is not connected ");  
        Console.WriteLine("to the system or is not turned on.");  
    }  
    if (e.ErrorCode == ErrorCode.Timeout)  
    {  
        Console.WriteLine("The Service Object timed out  
            waiting for a response from the POS device.");  
    }  
  
    // The example has not claimed the MSR, which is an  
    // exclusive-access device, before trying to enable  
    // it. This will throw the PosControlException  
    // and trigger the following conditional block.  
    // Once triggered, the MSR will be claimed and enabled.  
    if (e.ErrorCode == ErrorCode.NotClaimed)  
    {  
        Console.WriteLine("The POS application attempted to access ");  
        Console.WriteLine("an exclusive-use device that must be ");  
        Console.WriteLine("claimed before the method or property ");  
        Console.WriteLine("set action can be used.")  
        msr.Claim(1000);  
        msr.DeviceEnabled = true;  
    }  
}
```

See Also

Reference

- [PosException](#)
- [PosControlException](#)

- [PosManagementException](#)
- [PosLibraryException](#)

Concepts

- [PosExplorer Class](#)
- [POS Exception Handling](#)

Other Resources

- [POS for .NET API Support](#)

Event Handler Sample (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

This sample demonstrates how applications can register handlers for any of the five event types supported by Microsoft Point of Service for .NET (POS for .NET):

- DataEvent
- ErrorEvent
- StatusUpdateEvent
- OutputCompleteEvent
- DirectIOEvent

It also shows how an application manages asynchronous output requests to a [PosPrinter](#) device.

To create the sample project

1. Compile and install the Service Object sample code from [Asynchronous Output Sample](#).
2. Create a Windows Forms Application project in Microsoft Visual Studio 2013.
3. The code section below includes two files, `AsyncApp.cs` and `AsyncApp.Designer.cs`.
4. In the newly created project, replace `Form1.cs` with `AsyncApp.cs` and `Form1.Designer.cs` with `AsyncApp.Designer.cs`.
5. If it doesn't already exist, add an assembly reference to `Microsoft.PointOfService.dll`. In a standard install, you can find this file under `Program Files (x86)\Microsoft Point Of Service\SDK`.
6. Compile and run.

To run the sample

1. This sample displays a GUI user interface that allows the user to send asynchronous and synchronous print requests to the [Asynchronous Output Sample](#) Service Object.

2. The Service Object waits for several seconds before returning from a request, whether it is synchronous or asynchronous.

3. The UI displays the status of each request in the text box.

Example

This sample code demonstrates several key points:

- Using an **OutputCompleteEvent** event to notify the application that the Service Object has completed an output request.
- Registering event handlers, including using reflection to do so.
- Using [PosExplorer](#) to search for specific Service Objects.

To demonstrate how reflection can be used to discover which events are available on a given object, this code uses the [PosCommon](#) object returned from [CreateInstance\(DeviceInfo\)](#) without first casting it to a **PosPrinter**. In most cases, an application does not need to be generic in that way and so would cast the object as appropriate.

C#

```
// ASYNCAPP.CS
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.Reflection;
using Microsoft.PointOfService;

namespace AsyncOutputApp
{
    public partial class AsyncApp : Form
    {
        PosCommon posCommon;
        PosExplorer posExplorer;

        public AsyncApp()
        {
            InitializeComponent();

            btnPrintAsync.Enabled = true;
            btnPrintSync.Enabled = true;

            posExplorer = new PosExplorer(this);
        }
    }
}
```

```

posCommon = null;

string SName = "AsyncOutputPrinter";

try
{
    OpenDevice(SName);
    SetupEvents();
}
catch
{
    MessageBox.Show("The Service Object '" +
        SName + "' failed to load",
        "Service Object Error",
        MessageBoxButtons.OK,
        MessageBoxIcon.Exclamation);

    btnPrintAsync.Enabled = false;
    btnPrintSync.Enabled = false;
}
}

private void OpenDevice(string SName)
{
    string str = txtPrintResults.Text;
    DeviceInfo device = null;

    txtPrintResults.Clear();

    {
        // Retrieve the list of PosPrinter Service Objects.
        DeviceCollection devices =
            posExplorer.GetDevices(
                DeviceType.PosPrinter);

        // Iterate through the list looking for the one
        // needed for this sample.
        foreach(DeviceInfo d in devices)
        {
            if(d.ServiceObjectName == SName)
            {
                device = d;
                break;
            }
        }

        if (device == null)
        {
            throw new Exception("Service Object not found");
        }

        txtPrintResults.Text = "Opening device: " +
            device.ServiceObjectName + ", type: " +
            device.Type + "\r\n";
    }
}

```

```
posCommon =
    (PosCommon)posExplorer.CreateInstance(device);

    posCommon.Open();
    posCommon.Claim(0);
    posCommon.DeviceEnabled = true;
}
}

// When this button is pressed, AsyncMode is turned off,
// and the application waits for each print request
// to complete before regaining control.
private void btnPrintSync_Click(object sender, EventArgs e)
{
    PosPrinter posPrinter = posCommon as PosPrinter;
    posPrinter.AsyncMode = false;

    txtPrintResults.AppendText(
        "Printing will take place " +
        "synchronously.\r\n");

    StartPrinting();
}

// When this button is pressed, AsyncMode is turned on. Print
// requests will be queued and delivered on a
// first-in-first-out basis.
private void btnPrintAsync_Click(object sender, EventArgs e)
{
    PosPrinter posPrinter = posCommon as PosPrinter;
    posPrinter.AsyncMode = true;

    txtPrintResults.AppendText(
        "Printing will take place " +
        "asynchronously.\r\n");

    StartPrinting();
}

private void StartPrinting()
{
    PosPrinter posPrinter = posCommon as PosPrinter;

    txtPrintResults.AppendText(
        "Calling PrintNormal to start " +
        "printing...\r\n");

    // Notice that calling PrintNormal() here may not result
    // in a print request being sent immediately to the
    // printer. In asynchronous mode, the requested is
    // placed in a first-in-first-out queue managed by
    // POS for .NET.
    try
{
    posPrinter.PrintNormal(
```

```

                PrinterStation.Receipt,
                "This is do-nothing print data");
}
catch (PosControlException e)
{
    txtPrintResults.AppendText(
        "PrintNormal threw a " +
        "PosControlException! Description: " +
        e.Message + "\r\n");

    return;
}

// When data is sent to an output device, POS for .NET
// updates the OutputId property in the target object.
// When an OutputCompleteEvent is sent to the app,
// the OutputCompleteEventArgs will contain this id.
Int32 id = posPrinter.OutputId;

txtPrintResults.AppendText(
    "PrintNormal has returned! OutputID = " +
    id + "\r\n");
}

// Visual Studio-generated code.
private void AsyncApp_Load(object sender, EventArgs e)
{
}

#region Event Registration
private void SetupEvents()
{
    // All PosCommon objects support StatusUpdateEvent and
    // DirectIOEvent events, so simply register a handler
    // for those events.
    posCommon.StatusUpdateEvent +=
        new StatusUpdateEventHandler(
            co_OnStatusUpdateEvent);

    posCommon.DirectIOEvent +=
        new DirectIOEventHandler(
            co_OnDirectIOEvent);

    // In addition to the events common to all devices
    // (StatusUpdateEvent and DirectIOEvent), a device
    // type may also support DataEvent, ErrorEvent, or
    // OutputCompleteEvent events.
    //
    // In this example, the following code uses reflection
    // to determine which events are supported by this
    // object (posCommon).
    //
    // However, in the general case, an application will know
    // what type of device was returned when PosExplorer
    // CreateInstance() was called; therefore will not
}

```

```

// need to use reflection, but can instead register
// event handlers using the mechanism used for
// StatusUpdateEvent and DirectIOEvent events above.
EventInfo dataEvent =
    posCommon.GetType().GetEvent(
        "DataEvent");
if (dataEvent != null)
{
    dataEvent.AddEventHandler(posCommon,
        new DataEventHandler(
            co_OnDataEvent));

    txtPrintResults.AppendText("Registering Event: " +
        "DataEvent\r\n");
}

EventInfo errorEvent =
    posCommon.GetType().GetEvent(
        "ErrorEvent");
if (errorEvent != null)
{
    errorEvent.AddEventHandler(posCommon,
        new DeviceErrorHandler(
            co_OnErrorEvent));

    txtPrintResults.AppendText("Registering Event: " +
        "ErrorEvent\r\n");
}

EventInfo outputCompleteEvent =
    posCommon.GetType().GetEvent(
        "OutputCompleteEvent");
if (outputCompleteEvent != null)
{
    outputCompleteEvent.AddEventHandler(
        posCommon, new OutputCompleteEventHandler(
            co_OnOutputCompleteEvent));

    txtPrintResults.AppendText("Registering Event: " +
        "OutputCompleteEvent\r\n");
}
}

#endif Event Registration

#region Event Handlers
private void co_OnDataEvent(
    object obj,
    DataEventArgs d)
{
    txtPrintResults.AppendText(d.ToString() + "\r\n");
}

private void co_OnStatusUpdateEvent(
    object source,
    StatusEventArgs d)

```

```

        {
            txtPrintResults.AppendText(d.ToString() + "\r\n");
        }

        private void co_OnDirectIOEvent(
            object source,
            DirectIOEventArgs d)
        {
            txtPrintResults.AppendText(d.ToString() + "\r\n");
        }

        private void co_OnErrorEvent(
            object source,
            DeviceErrorEventArgs d)
        {
            string str = d.ToString();

            MessageBox.Show(d.ToString(),
                "OnErrorEvent called",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);

            txtPrintResults.AppendText(d.ToString() + "\r\n");
        }

        private void co_OnOutputCompleteEvent(
            object source,
            OutputCompleteEventArgs d)
        {
            txtPrintResults.AppendText(d.ToString() + "\r\n");
        }
    #endregion Event Handlers
}

}

// ASYNCAPP.DESIGNER.CS
namespace AsyncOutputApp
{
    partial class AsyncApp
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
        }
    }
}

```

```
        }

        base.Dispose(disposing);
    }

#region Windows Form Designer generated code

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.txtPrintResults = new System.Windows.Forms.TextBox();
    this.btnPrintSync = new System.Windows.Forms.Button();
    this.btnPrintAsync = new System.Windows.Forms.Button();
    this.SuspendLayout();
    //
    // txtPrintResults
    //
    this.txtPrintResults.BackColor =
System.Drawing.SystemColors.Window;
    this.txtPrintResults.Location = new System.Drawing.Point(12,
119);
    this.txtPrintResults.Multiline = true;
    this.txtPrintResults.Name = "txtPrintResults";
    this.txtPrintResults.ReadOnly = true;
    this.txtPrintResults.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
    this.txtPrintResults.Size = new System.Drawing.Size(650, 200);
    this.txtPrintResults.TabIndex = 3;
    //
    // btnPrintSync
    //
    this.btnPrintSync.Location = new System.Drawing.Point(12, 12);
    this.btnPrintSync.Name = "btnPrintSync";
    this.btnPrintSync.Size = new System.Drawing.Size(132, 39);
    this.btnPrintSync.TabIndex = 1;
    this.btnPrintSync.Text = "Print Synchronous";
    this.btnPrintSync.UseVisualStyleBackColor = true;
    this.btnPrintSync.Click += new
System.EventHandler(this.btnPrintSync_Click);
    //
    // btnPrintAsync
    //
    this.btnPrintAsync.Location = new System.Drawing.Point(12, 57);
    this.btnPrintAsync.Name = "btnPrintAsync";
    this.btnPrintAsync.Size = new System.Drawing.Size(132, 39);
    this.btnPrintAsync.TabIndex = 2;
    this.btnPrintAsync.Text = "Print Asynchronously";
    this.btnPrintAsync.UseVisualStyleBackColor = true;
    this.btnPrintAsync.Click += new
System.EventHandler(this.btnPrintAsync_Click);
    //
    // AsyncApp
    //
}
```

```

        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(685, 331);
        this.Controls.Add(this.btnPrintAsync);
        this.Controls.Add(this.btnPrintSync);
        this.Controls.Add(this.txtPrintResults);
        this.Name = "AsyncApp";
        this.Text = "Form1";
        this.Load += new System.EventHandler(this.AsyncApp_Load);
        this.ResumeLayout(false);
        this.PerformLayout(false);
        this.PerformLayout();

    }

}

private System.Windows.Forms.TextBox txtPrintResults;
private System.Windows.Forms.Button btnPrintSync;
private System.Windows.Forms.Button btnPrintAsync;
}
}

```

The program should display the following text if you first press the **Print Synchronous** and then the **Print Asynchronous** button.

Opening device: AsyncOutputPrinter, type: PosPrinterRegistering Event:
ErrorEventRegistering Event: OutputCompleteEventPrinting will take place
synchronously.Calling PrintNormal to start printing...PrintNormal has returned! OutputID
= 0Printing will take place asynchronously.Calling PrintNormal to start
printing...PrintNormal has returned! OutputID =
1Microsoft.PointOfService.OutputCompleteEventArgs, TimeStamp: 11:35:39 AM, EventId:
1, OutputId: 1.

See Also

Tasks

- [Asynchronous Output Sample](#)

Concepts

- [Event Management](#)

Other Resources

- Developing a Custom Service Object

Developing a Custom Service Object

(POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The POS for .NET SDK delivers support for developing UPOS-compliant applications. The POS for .NET class tree delivers support for both applications that use a specific UPOS device, and Service Objects which provide the link between the physical hardware and the application.

Service Objects are typically implemented by independent hardware vendors (IHVs) to provide an interface between POS for .NET applications and the IHWs particular UPOS device. The POS for .NET SDK includes a set of .NET classes which you can use to create fully functional, UPOS-compliant Service Objects. By taking advantage of the POS for .NET SDK classes, you can write Service Objects quickly and with relatively little device-specific code.

In This Section

- [POS for .NET Service Object Architecture](#) Presents an overview of the POS for .NET architecture used for building custom Service Objects.
- [System Configuration](#) Provides information about settings and configuration for POS for .NET installations.
- [Service Object Samples: Getting Started](#) Provides a step-by-step guide to creating a functional, multithreaded Service Object.
- [Developing Service Objects Using Base Classes](#) POS for .NET includes a nearly complete Base class implementation for nine POS device types. This section explains how to use these classes as a foundation for device-specific Service Objects.
- [Device Input and Events](#) Explains how events are used to send input from the POS device to the application.
- [Device Output Models](#) Explains the difference between synchronous and asynchronous output to POS devices.
- [Asynchronous Output Sample](#) Implements a simple `PosPrinterBase` Service Object in order to demonstrate how POS for .NET manages asynchronous output.

- [Statistics Sample](#) Implements a sample Service Object that uses the POS for .NET statistics methods.
- [Base Class DirectIO Method](#) Explains how Service Objects can implement the method **DirectIO** to provide access to manufacturer-specific data to the application.
- [Capability Properties](#) Demonstrates how certain properties, such as capability properties, are set by the Service Object.

Related Sections

- [POS for .NET v1.14 Features](#) Provides a high-level overview of the POS for .NET system.
- [Developing a POS Application](#) Provides details for creating a POS application using POS for .NET.
- [POS Device Manager](#) Describes using POS Device Manager to configure a POS for .NET installation.

POS for .NET Service Object Architecture (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

There are features and design elements in POS for .NET that must be taken into account by Service Object developers. This section includes topics discussing several of these important concepts.

In This Section

- [POS for .NET Class Tree](#) Provides an overview of the POS for .NET class structure. POS for .NET Service Object classes are derived from an appropriate class in this tree.
- [Attributes for Identifying Service Objects and Assigning Hardware](#) Explains how POS for .NET uses reflection and .NET attributes to locate POS for .NET assemblies and identify valid Service Objects within those assemblies.
- [Hydra Devices](#) Provides a broad overview of the problems involved when developing for hardware that may be in use by several Service Objects simultaneously.
- [Plug and Play Support](#) Provides an overview of Plug and Play support available to Service Object developers.
- [PosCommon Information for Service Object Developers](#) Describes methods and properties of `PosCommon` that are useful to Service Object developers.
- [POS for .NET Device Basic Classes](#) Provides information about using the POS for .NET **Basic** classes as the starting point for Service Objects.

Related Sections

- [POS for .NET Architecture](#) Describes the components that support developers writing POS applications as well as peripheral device hardware vendors writing .NET-based Service Objects.
- [Typical POS Application Architecture](#) Describes event handlers and the public interface to Service Objects and the `PosExplorer Class` as the principal elements of

POS application architecture.

- [Service Object Samples: Getting Started](#) Includes a step-by-step guide to creating a functioning Service Object and demonstrates how the features discussed above are utilized.
- [Developing a POS Application](#) Explains how to create POS for .NET applications, the ultimate consumers of POS for .NET Service Objects.
- [System Configuration](#) Describes how to configure POS for .NET to meet the requirements of your installation.

POS for .NET Class Tree (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The POS for .NET SDK contains a set of classes which provide the Service Object with much of the functionality needed to meet the UPOS specification. There are three levels of base classes, referred to as **Interface**, **Basic**, and **Base** classes.

At the base of the class tree is [PosCommon Class](#). **Interface** classes are derived from **PosCommon**, **Basic** classes are derived from **Interface** classes, and **Base** classes are derived from **Basic** classes. For each POS device type, there are separate **Interface**, **Basic**, and **Base** classes.

The POS for .NET base classes follow a specific naming convention. **Interface** classes are represented by just the short name of the device type (for example, **Scanner** or **Msr**). **Basic** classes append the suffix **Basic** after the name used for the **Interface** class (for example, **MsrBasic** or **ScannerBasic**). And finally, **Base** classes use the suffix **Base** (for example, **MsrBase** or **ScannerBase**). For the complete list of class names, see [Supported Device Classes](#).

Interface Classes

The **Interface** classes are the most fundamental base classes provided by POS for .NET. There is an **Interface** class for each of the 36 device types in the UPOS specification, and they contain methods and properties that correspond to those required by the specification. They provide no device-specific functionality so deriving from these classes requires the Service Object developer to provide the greatest amount of additional code and therefore should rarely be used directly.

Basic Classes

Basic classes are derived from their corresponding **Interface** class. There is a **Basic** class for all 36 devices supported by the UPOS specification. These classes provide some functionality and are the best choice if no **Base** class exists for your device type. **Basic** classes, however, implement only the UPOS common members.

Base Classes

The **Base** classes, each of which is derived from its corresponding **Basic** class, offer the greatest level of functionality. The **Base** classes provide nearly complete Service Object implementations. By deriving from these classes, the Service Object developer only needs to implement code to control the specific hardware device. Since **Base** classes provide so much functionality, Service Object developers should use them whenever possible. POS for .NET provides **Base**-level support for only nine **primary** device types.

UPOS Device	Corresponding POS for .NET Base Class
Cash Drawer	CashDrawerBase
Check Scanner	CheckScannerBase
Line Displays	LineDisplayBase
Magnetic Stripe Reader	MsrBase
Pin Pad	PinPadBase
POS Keyboards	PosKeyboardBase
POS Printers	PosPrinterBase
RFIDScanner	RFIDScanner
Scanner (Bar Code Reader)	ScannerBase

See Also

Concepts

- [Supported Device Classes](#)

Other Resources

- [Developing a Custom Service Object](#)
- [POS for .NET Service Object Architecture](#)

Attributes for Identifying Service Objects and Assigning Hardware (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

POS for .NET uses .NET reflection and .NET attributes to locate Service Object assemblies, identify Service Objects within those assemblies, and finally to associate a Plug and Play device with that Service Object. By leveraging these .NET features, [PosExplorer](#) can identify Service Objects within an assembly and quickly assess their Plug and Play requirements. The expensive process of loading a .NET assembly is delayed until needed by the application.

In order to provide these features, POS for .NET depends on three different .NET attributes:

- **PosAssembly** This is a global, assembly-level attribute that tells [PosExplorer](#) that this is a POS for .NET assembly which contains one or more Service Objects. Generally, it should be set in your `AssemblyInfo.cs` source file. For an example, see [Setting up a Service Object Project](#).
- **ServiceObject** This attribute is applied to the Service Object class and specifies the type, name, and version information for the Service Object. See the [Creating a Basic Service Object Code Template](#) section for an example.
- **HardwareId** This attribute is used to specify which hardware IDs will be used by this Service Object. This information is used by [PosExplorer](#) to filter out Service Objects that use Plug and Play hardware which is not currently plugged in. The **HardwareId** attribute allows multiples, so there may be several attached to a Server Object class. See the sample topic [Adding Plug and Play Support](#) for an example. For a more lengthy discussion of Plug and Play features, including how the **HardwareId** attribute is utilized, see the topics [Adding Plug and Play Support](#) and [POS for .NET Integration with Plug and Play](#).

See Also

Reference

- [PosAssemblyAttribute](#)
- [HardwareIdAttribute](#)
- [ServiceObjectAttribute](#)

Concepts

- [Plug and Play Support](#)
- [POS for .NET Registry Settings](#)

Other Resources

- [POS for .NET Service Object Architecture](#)
- [System Configuration](#)
- [Service Object Samples: Getting Started](#)

Hydra Devices (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Some peripheral POS devices combine UPOS device types. These are referred to as Hydra devices, and their interface to the POS application requires more than one Service Object.

For example, a magnetic ink character recognition (MICR) device may include a POS printer. In that case, the device is represented by both a MICR Service Object and a POS printer Service Object. Even though they interact with the same peripheral device, both Service Objects must be created and controlled separately. The MICR Service Object manages the MICR check scanning and character recognition function and the POS printer Service Object manages the receipt and validation printers.

However, both MICR and POS printer Service Objects must work together in a single transaction. Check processing combines check insertion and removal operations in the MICR device with validation printing functions in the POS printer.

Considerations

In the normal case, a Service Object would simply open a connection to the device and perform its read and write operations. With **Hydra** devices, however, the task is more complicated since IO ports are normally exclusive. Therefore, multiple Service Objects accessing the same device must synchronize with each other, typically with some variety of inter-process communication.

POS for .NET offers no features to help multiple Service Objects synchronize with each other. The Service Object developer must write this code and tailor it to the specific system configuration.

See Also

Concepts

- [Service Object Overview](#)

Other Resources

- POS for .NET Service Object Architecture

Plug and Play Support (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

POS for .NET fully supports Windows Embedded Plug and Play technology. To utilize this feature, Service Object developers can simply add one or more [HardwareId](#) attributes to their code, or include the hardware reference in a [Plug and Play XML Configuration](#) file.

Adding this attribute to a Service Object helps application developers, who will now know that when they use [PosExplorer](#) to get a list of Service Objects, any Service Object in that list will be associated with a functioning POS device. The application benefits directly from this association by greater reliability and ease of use. We recommend that Service Objects support the Plug and Play feature whenever possible.

Plug and Play Behavior

Once the Service Object has been associated with the POS device's hardware ID, POS for .NET uses the Windows Plug and Play Manager to determine what POS devices are connected to the computer. No additional code is required by the application or Service Object.

When an application invokes the method [PosExplorer.GetDevices](#), [PosExplorer](#) finds the device that is associated with each Plug and Play Service Object and then queries the Windows Plug and Play Manager to determine the device's status. If the device is not available, it will not be added to the device list that is returned to the application from [PosExplorer.GetDevices](#).

PosExplorer Service Object Filtering

[PosExplorer](#) is able to effectively filter the list of Plug and Play Service Objects when the application calls [PosExplorer.GetDevices](#). The filtering process works as follows:

1. Searches for all assemblies in the specified POS for .NET directories.
2. If the assembly is not marked with the [PosAssembly](#) global attribute, discards it.
3. Searches for classes marked with the [ServiceObject](#) attribute. For each such class:
 - a. Looks for a hardware ID associated with this class, as either a [HardwareId](#) attribute or within the [Plug and Play XML Configuration](#) file. If there is no hardware ID, leaves the Service Object in the [PosExplorer](#) list.

- b. If there is a hardware ID, then queries Windows to retrieve the device's status. If the device is attached to the computer, leaves it in the **PosExplorer** list.
- c. If the device is not attached to the computer, removes it from the **PosExplorer** list.

Example

The following code example demonstrates a simple method of handling Plug and Play events. Information generated by **PosExplorer** is used to instantiate the correct device, in this case a Magnetic Stripe Reader (MSR).

C#

```
// Connect the Plug and Play events to detect the removal or
// connection of a new device.
explorer.DeviceAddedEvent += new
    DeviceChangedEventHandler(explorer_DeviceAddedEvent);
explorer.DeviceRemovedEvent += new
    DeviceChangedEventHandler(explorer_DeviceRemovedEvent);

// This event handler extends Plug and Play functionality to the MSR
// device type. A message is printed to the console if the connection
// is successful.
void explorer_DeviceAddedEvent(object sender, DeviceChangedEventArgs e)
{

    // Checks if the newly added device is an MSR.
    if (e.Device.Type == DeviceType.Msr)
    {

        // Checks if an MSR instance has already been created and,
        // if not, creates one. If a new MSR instance is created, its
        // name is recorded in a string and written to the console.
        // Once the printing is finished, the MSR is closed.
        if (msr == null)
        {
            CreateMsr(e.Device);
            strMsrConfig = e.Device.ServiceObjectName;
            Console.WriteLine(strMsrConfig);
            // It is important that applications close all open
            // Service Objects before terminating.
            msr.Close();
        }
    }
}
```

See Also

Tasks

- [Adding Plug and Play Support](#)

Reference

- [PosExplorer](#)
- [HardwareIdAttribute](#)
- [PosAssemblyAttribute](#)
- [ServiceObjectAttribute](#)

Concepts

- [Attributes for Identifying Service Objects and Assigning Hardware](#)
- [POS for .NET Registry Settings](#)
- [Plug and Play XML Configuration](#)

Other Resources

- [POS for .NET Service Object Architecture](#)

PosCommon Information for Service Object Developers (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

At the base of the POS for .NET Server Object class tree ([Interface](#), [Basic](#), [Base](#)) is [PosCommon](#). This class is a direct implementation of the "Common Properties, Methods, and Events" chapter in the UPOS specification.

Each POS for .NET **Basic** class overrides or implements **PosCommon** properties and methods which are of particular importance to a Service Object developer. This topic provides information about these methods and properties.

CapPowerReporting Property

Once POS for .NET has successfully opened a device, it tries to retrieve the current value of the [CapPowerReporting](#) property. The [PowerReporting](#) class is initialized to **None**, indicating that the Service Object is not able to provide power reporting. If, however, the Service Object's device does support power reporting, the Service Object may set [PowerReporting](#) to **Standard** or **Advanced** in the Service Object's method.

CapStatisticsReporting Property

POS for .NET verifies that the device has been opened and then retrieves the current value of the [CapStatisticsReporting](#) property.

When statistics are created for the device, POS for .NET sets [CapStatisticsReporting](#) to **true**.

CapUpdateStatistics Property

POS for .NET verifies that the device has been opened and then retrieves the current value of the [CapUpdateStatistics](#) property.

When statistics are created for the device, and if those statistics can be reset or updated, then POS for .NET sets [CapUpdateStatistics](#) to **true**.

Claimed Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the [Claimed](#) property.

Claimed is initialized to **false**. **Claimed** should be set to **true** when the application calls the [Claim\(Int32\)](#) method, then set back to **false** when the application calls the [Release\(\)](#) method.

DeviceDescription Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the [DeviceDescription](#) property.

DeviceEnabled Property

[DeviceEnabled](#) is a read/write property.

It can be used to return the object's current state; enabled or disabled. If this object has not been previously opened and enabled, this property returns **false**.

This property is also used to enable or disable the device by setting the property of the value to **true** or **false**. It is common for Service Objects to override this property and perform its hardware initialization and release here.

DeviceName Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the [DeviceName](#) property.

Within **Base** class implementations, this value is set automatically based on the contents of the [ServiceObject](#) attribute.

If you are not deriving from a POS for .NET **Base** class, and are instead deriving from an **Interface-level** or **Basic-level** class, then **DeviceName** should be set by the Service Subject during the [Open\(\)](#) method.

FreezeEvents Property

[FreezeEvents](#) is a read/write property.

POS for .NET verifies that the device has been opened and claimed, then retrieves or sets the current value of the **FreezeEvents** property. When this property is set to **true**, POS for .NET queues events until this property is set to **false**, not that the queuing mechanism can vary from one device type to another.

The **FreezeEvents** property is initialized to **false**.

PowerNotify Property

PowerNotify is a read/write property.

POS for .NET verifies that the device has been opened, then retrieves or sets the current value of **PowerNotify**. If **PowerNotify** is set, then power state notifications will be sent to the application.

PowerNotify is initialized to **Disabled**.

Attempting to set **PowerNotify** can cause the following exceptions to be thrown.

Value	Meaning
Illegal	One of the following conditions has occurred: <ul style="list-style-type: none">• The device is enabled.• P:Microsoft.PointOfService.PosCommon.CapPowerReporting is set to None, indicating that the device does not support power notification.• The specified value is not a valid T:Microsoft.PointOfService.PowerNotification enumeration value.

PowerState Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the **PowerState** property. If **CapPowerReporting** is set to **None**, **PowerNotify** is set to **Disabled**, or **DeviceEnabled** set to **false**, **PowerState** is returned as **Unknown**.

PowerState is initialized to **Unknown**. When **PowerNotify** is set to **Enabled** and **DeviceEnabled** is **true**, **PowerState** should be updated as the Service Object detects power condition changes. POS for .NET detects the state change when the Service Object sets **PowerState** and—if **PowerNotify** is set to **Enabled**—queues a **StatusUpdateEvent** event, notifying the application.

Setting **PowerState** can cause the following exceptions to be thrown.

Value	Meaning
Illegal	<p>One of the following conditions has occurred:</p> <ul style="list-style-type: none"> • CapPowerReporting = Standard and PowerNotify is set to Online, Off, or Offline. • CapPowerReporting = Advanced and PowerState is set to Online, Off, or Offline.

ServiceObjectDescription Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the [ServiceObjectDescription](#) property. The Service Object developer should not have to set this value, since it is set by the POS for .NET **Basic** class using the description information provided in the [ServiceObject](#) attribute.

ServiceObjectVersion Property

POS for .NET verifies that the device has been opened, and then retrieves the current value of the [ServiceObjectVersion](#) property. The Service Object developer should not have to set this value, since it is set by the POS for .NET **Basic** class using the version information provided in the [ServiceObject](#) attribute.

State Property

No device state verification is required—the application can retrieve the current value of the [State](#) property at any time.

State is initialized to **Closed**. If the Service Object sets **State** to an invalid [ControlState](#) value, POS for .NET throws an **Illegal** exception. Changes in **State** cause POS for .NET to queue a **StateChangedEvent** event.

Claim Method

POS for .NET verifies that the application has the device opened.

If the *timeout* parameter is set to a value less than -1, POS for .NET throws an exception. If the *timeout* value is set to -1, the **Claim** method will wait forever.

If the device is already claimed by the application, POS for .NET simply returns.

If the device is in use by another application, *timeout* is reached; POS for .NET throws a time-out exception.

If **Claim** is successful, POS for .NET loads the statistics for the device and sets the **Claimed** property to **true**.

Close Method

If the application calls the [Close\(\)](#) method when **State** is set to **Closed**, POS for .NET throws a **Closed** exception. If **State** is set to **Busy**, POS for .NET throws a **Busy** exception.

If the **DeviceEnabled** method when **Claimed** is set to **false**, POS for .NET throws an **Illegal** exception. If **State** is set to **Busy**, POS for .NET calls the **ClearOutput** method. If the device is enabled, POS for .NET sets **DeviceEnabled** to **false**. POS for .NET clears the event queue, and then sets **Claimed** to **false**.

ResetStatistic Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [ResetStatistic\(String\)](#) method.

ResetStatistics() Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [ResetStatistics\(\)](#) method.

ResetStatistics(categories parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [ResetStatistics\(StatisticCategories\)](#) method.

ResetStatistics(string parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [ResetStatistics\(String\[\]\)](#) method.

RetrieveStatistic Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [RetrieveStatistic\(String\)](#) method.

RetrieveStatistics() Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [RetrieveStatistics\(\)](#) method.

RetrieveStatistics(categories parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [RetrieveStatistics\(StatisticCategories\)](#) method.

RetrieveStatistics(string parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [RetrieveStatistics\(String\[\]\)](#) method.

UpdateStatistic Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [UpdateStatistic\(String, Object\)](#) method.

UpdateStatistics(categories parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [UpdateStatistics\(StatisticCategories, Object\)](#) method.

UpdateStatistics(statistic array parameter) Method

POS for .NET verifies that the application has opened, claimed, and enabled the device, then calls the [UpdateStatistics\(Statistic\[\]\)](#) method.

See Also

Reference

- [PosCommon](#)

Concepts

- POS for .NET Class Tree

Other Resources

- Service Object Samples: Getting Started

POS for .NET Device Basic Classes (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Each hardware device in POS for .NET is represented by both an abstract interface, such as [CashDrawer](#) class, and a **Basic** class, such as [CashDrawerBasic](#). **Basic** classes derive from the underlying interface and contain basic functional support for the device. POS for .NET provides generic support for opening, claiming, and enabling the device, device statistics, and for managing delivery of events to the application. In addition, each **Basic** class contains a set of inherited and protected methods that can be implemented by the Service Object. This topic provides summary information about **Basic** classes that can be used by Service Objects that derive from the device's **Basic** class, rather than taking advantage of the more fully implemented device **Base** class.

Constructor

Each **Basic** class includes a constructor that creates an instance of the class and initializes statistics for the UPOS version, the device category, and the installation date.

Common Properties and Methods

Each **Basic** class provides overridden [PosCommon](#) property and method definitions. For each of these properties and methods, the **Basic** class handles state validation—that is, verification that the application has opened, claimed, or enabled the device—and then calls the POS for .NET implementation of that property or method. For more information about the **PosCommon** class, see [PosCommon](#).

The Service Object can use the [CommonProperties](#) class to update [PosCommon](#) properties that are designated read-only for the application, or to update those properties without worrying about state validation.

Dispose Methods

Each **Basic** class includes two implemented **Dispose** methods for use by the Service Object. For information about how these work, see the .NET Framework documentation for the [IDisposable](#) class.

Opening, Claiming, and Enabling Devices

Each **Basic** class provides the core functionality for opening, claiming, and enabling devices. Typically, though, Service Objects want to override these methods to add their own custom processing.

Protected Methods and Events for Service Object Developers

Each **Basic** class contains a group of methods and events for the Service Object developer.

The following protected properties are defined as follows:

- **CommonProperties** property, which returns an instance of **CommonProperties** with get and set values for all **PosCommon** properties. The Service Object can use **CommonProperties** to update properties without worrying about state validation or whether the property is designated read-only for the application.
- **ExternallyClaimed** property, which Service Objects can retrieve to determine if another instance of the device has been claimed (in which case, the property is set to **true**).
- **ErrorCount** property. When the Service Object queries for the value of **ErrorCount**, the basic class checks the event queue and tallies the number of **ErrorEvent** events found, and then returns that tally as the value of **ErrorCount**.
- **DataCount** property. POS for .NET verifies that the device has been opened, and then returns the number of **DataEvent** events currently queued for the device.

The following protected methods are defined:

- **StateChangedEvent** and delegate **StateChangedEventHandler (EventArgs class)**. The Service Object can implement these to receive notification when the device's **State** property has changed.
- **PreFireEvent** protected methods for each type of event supported by the device. Each basic class provides a default, generic implementation of **PreFireEvent** that returns immediately. If the Service Object needs to update its internal state prior to an event being sent to the application, the Service Object can override the default implementation of **PreFireEvent** and provide its own implementation for the event type in question.
- **QueueEvent** protected methods for each type of event supported by the device. The Service Object calls **QueueEvent** to add an event to the event queue. The **Basic** class verifies that the device is enabled, and then adds the event to the event

queue to be delivered to the application. Immediately before delivery, the **Basic** class calls the appropriate **PreFireEvent** to give the Service Object an opportunity to update its internal state. When **PreFireEvent** returns, the **Basic** class delivers the event to the application.

- **QueueEventAndWait** protected methods. The Service Object calls **QueueEventAndWait** to add an **ErrorEvent** event or **DirectIOEvent** event to the event queue, from which the Service Object expects a response from the application. The **Basic** class verifies that the device is enabled, and then adds the event to the event queue, to be delivered to the application when conditions are correct. Immediately before delivery, the **Basic** class calls **PreFireEvent** to give the Service Object an opportunity to update its internal state. When **PreFireEvent** returns, the **Basic** class delivers the event to the application.
- **VerifyState** method, which takes two Boolean values, *mustBeClaimed* and *mustBeEnabled*. The Service Object can call the POS for .NET implementation of this method to perform the necessary state validation for the device, prior to a method or property call.
- **CreateStatistic** method. The Service Object should use these methods to create custom (that is, manufacturer-specific) statistics. POS for .NET handles the creation and management of all UPOS-defined statistics.
- **SetStatisticValue** and **IncrementStatistic** methods allow the Service Object to update a specified statistic even if it is not defined as resettable (that is, these methods bypass the rules enforced by the [PosCommonResetStatistic\(String\)](#) and [UpdateStatistic\(String, Object\)](#) methods).
- **SetStatisticHandlers(String, GetStatistic, SetStatistic)** method, which allows Service Objects to provide external callback functions for the retrieval and setting of hardware-based statistics. If a get property is not defined, the **Basic** class assumes that the statistic is software-based, and its value is maintained in the statistics XML file. If both get and set properties are defined, the **Basic** class calls these functions whenever the statistic needs to be updated or reset. It is up to the Service Object to provide code to update the statistic in the hardware.
- **ClearInput** method. POS for .NET clears the event queues for the device and then calls the Service Object's implementation of **ClearInputImpl**.
- **ClearInputImpl** method. The Service Object should implement this method to clear any hardware buffers for the device.

See Also

Other Resources

- [Developing a Custom Service Object](#)

- POS for .NET Service Object Architecture

System Configuration (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) has several configurable settings which can be adapted to meet the needs of a particular installation. This section explains how to modify the various POS for .NET settings.

In This Section

- [POS for .NET Registry Settings](#) Shows which keys and values are used by POS for .NET and how they can be modified. Basic POS for .NET settings are stored in the Windows Registry.
- [POS Device Manager Output](#) Contains details about how to use the configuration file created by the POS Device Manager (POSDM.EXE).
- [Point of Service Performance Counters](#) Explains how to use POS for .NET performance counters to monitor your system.
- [Plug and Play XML Configuration](#) Demonstrates how POS devices can be associated with a specific Service Object using an XML file instead of assembly attributes.

Related Sections

- [Service Object Samples: Getting Started](#) Includes a step-by-step guide to creating a functioning Service Object and demonstrates how the features discussed above are utilized.
- [POS for .NET Service Object Architecture](#) Outlines the general concepts of the POS for .NET architecture.

POS for .NET Registry Settings (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) stores certain configuration information in the system registry. During installation, default values are written to the registry. The POS for .NET values are stored under the key `\HKLM\SOFTWARE\POSfor.NET`. Below is a list of registry keys and their values that POS for .NET uses.

POSfor.NET Key

This key contains the following values.

Name	Description	Data Type	Default value
Configuration	Name of the configuration file written by the POS Device Manager.	REG_SZ	C:\Documents and Settings\All Users\Application Data\Microsoft\Point Of Service\Configuration\Configuration.xml
StatisticsFile	Name of the file used to record POS for .NET statistics.	REG_SZ	C:\Documents and Settings\All Users\Application Data\Microsoft\Point Of Service\Statistics\PosDeviceStatistics.xml

The **POSfor.NET** registry key has three subkeys:

- ControlAssemblies
- ControlConfigs
- Logging

POSfor.NET\ControlAssemblies Key

This key may contain any number of values of type REG_SZ, each of which contains the name of a directory. [PosExplorer](#) will iterate through the entire list of values, searching each directory. Therefore, the names of the values are not important.

These values will need to be modified during system configuration so that they point to the locations appropriate for the specific requirements of the installation.

The following table shows the default values that are written during the POS for .NET SDK setup process.

Name	Default Values
(Default)	C:\Program Files\Common Files\Microsoft Shared\Point Of Service\Control Assemblies\
ExampleSOs	C:\Program Files\Microsoft Point Of Service\SDK\Samples\Example Service Objects\
Simulators	C:\Program Files\Microsoft Point Of Service\SDK\Samples\Simulator Service Objects\

POSfor.NET\ControlConfigs Key

In most cases, POS devices are paired with specific Service Objects using the [HardwareId](#) attribute, but in some rare situations, a Service Object provider may need the ability to assign a different device to a Service Object without redistributing the entire assembly.

To accommodate these situations, POS for .NET supports the ability to associate the device to a Service Object in a [Plug and Play XML Configuration](#) file.

This key contains a value which points to the location of these Plug and Play configuration files.

Name	Default Value
(Default)	C:\Program Files\Common Files\Microsoft Shared\Point Of Service\Control Configurations\

POSfor.NET\Logging Key

This key contains values which dictate how POS for .NET handles log files. Both POS for .NET and applications, using the [Logger](#) object, may write to the log file.

The following table shows the values of this key.

Name	Description	Data Type	Default
Enabled	Set to true if logging is enabled.	REG_DWORD	0 (not enabled)
Location	The location where the log files will be written.	REG_SZ	%TEMP%

Name	Description	Data Type	Default
MaxLogFileSizeMB	The maximum allowed log size, in megabytes.	REG_DWORD	10
Name	The Base name of the log file. Date and time information follows the file name. The .txt extension is appended.	REG_SZ	CCL

See Also

Reference

- [Logger](#)

Concepts

- [Log Files](#)
- [Plug and Play XML Configuration](#)
- [Plug and Play Support](#)

Other Resources

- [System Configuration](#)

POS Device Manager Output (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The [POS Device Manager](#) is used to set up the device configuration for a given system. The output from the **POS Device Manager** is written to an XML file. The name of this file is the value of the registry key `\HKLM\SOFTWARE\POSfor.NET\Configuration`.

Configuration Migration

You should not modify this file directly. Doing so may lead to unexpected behavior. You can, however, migrate the file from one system to another, making it possible to build a configuration on one system, and propagate that same setup to others.

API Support

You may also access configuration properties with the following methods in [PosCommon](#):

- [GetConfigurationProperty\(String\)](#)
- [SetConfigurationProperty\(String, String\)](#)
- [DeleteConfigurationProperty\(String\)](#)

See Also

Other Resources

- [POS Device Manager](#)

Point of Service Performance Counters (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The Point of Service Performance Counters service application helps applications and management tools to monitor device statistics. It is included in the standard Microsoft Point of Service for .NET (POS for .NET) Package Installer. However, it is disabled by default.

Device statistics are supported by POS for .NET **Basic** classes. They contain the code used by Service Object developers to implement statistics. Device statistic values are periodically saved to an XML file where they can be persisted across sessions. The POS for .NET statistics service uses this XML file to expose the statistics as performance counters that can be monitored by management tools and do not require direct interaction with a Service Object. Even if the Service Object is claimed by another POS application, the counters can still be monitored by other tools.

The service application creates a counter for any statistic that is represented by a value that may be cast to an integer.

To enable Point of Service performance counters

1. On the taskbar, choose **Start**, choose **Control Panel**, and then double-click **Administrative Tools**.
2. Choose **Services** to open the **Services** window.
3. Right-click the **Point of Service Performance Counters** service and choose **Properties**.
4. In the **Point of Service Performance Counters Properties** window, open the **Startup Type** drop-down menu and select when to enable Point of Service Performance Counters. There are three options:
 - **Disabled**. This is the default option. When Performance Counters are disabled, the service cannot run.
 - **Manual**. When Performance Counters are set to **Manual**, the service may be run, but does not automatically do so.

- **Automatic.** When Performance Counters are set to **Automatic**, the service automatically runs every time that your computer starts or restarts.
5. If you have selected **Manual** or **Automatic** in the **Startup Type** drop-down menu, start the service by choosing **Start the Service**.

To monitor Point of Service performance counters

1. On the taskbar, choose **Start**, choose **Control Panel**, and then choose **Administrative Tools**.
2. Choose **Performance** to open the **Performance Monitor**.
3. Right-click the **Performance** window and choose **Add Counters**.
4. In the **Add Counters** window, open the **Performance Object** drop-down list and select the POS for .NET device class whose statistics you want to display.
5. Select the counters associated with the device class that you have chosen to monitor. Then choose **Add**.
6. Choose **Close** to return to the **Performance** window.

See Also

Tasks

- [Statistics Sample](#)

Plug and Play XML Configuration (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Although the Plug and Play hardware ID should generally be specified using the **HardwareId** property within the Service Object source code, there may be times when Service Object vendors require more flexibility. For example, the hardware ID needs to be changed without redistributing the entire Service Object assembly.

To support these cases, Microsoft Point of Service for .NET (POS for .NET) specifies hardware associations in an XML file. These XML files are read from the directory that is specified in the registry key

HKEY_LOCAL_MACHINE/SOFTWARE/POSfor.NET/ControlConfigs. When constructing the list of available Service Objects and devices, [PosExplorer](#) processes each file in that directory and associates the device where possible. No additional action is required by either the Service Object or the application.

Schema

A Plug and Play configuration file must begin with a top-level node named **PointOfServiceConfig** and have the attribute **Version** to indicate the XML version of the file.

Following that, there may be any number of **ServiceObject** subnodes. Each service object node must include **Type** and **Name** attributes to indicate the POS device type and name of the Service Object. These two fields will be matched against available Service Objects to determine which, if any, should be associated with devices specified in the subnode **HardwareId**. There is also an optional attribute on the **ServiceObject** node, **Override**. If this attribute is set, then the device associations in the XML file overrides those contained in the assembly.

The **ServiceObject** node contains subnodes with the name **HardwareId**, which have **From** and **To** attributes. The contents of these attributes are the same as would be found in the **HardwareId** attribute in a Service Object assembly and specify the range of hardware IDs to associate with the Service Object.

Example

The example shows a typical XML Plug and Play configuration file.

XML

```
<PointOfServiceConfig Version="1.0">
    <ServiceObject Type="Msr" Name="ExampleMsr" Override="yes">
        <HardwareId From="HID\Vid_0801&Pid_0002&Rev_0100"
                     To="HID\Vid_0801&Pid_0002&Rev_9999" />
    </ServiceObject>
</PointOfServiceConfig>
```

Hardware ID Precedence

If the **Override** attribute on the **ServiceObject** node is set, then the device association specified in the XML takes precedence, and any **HardwareId** attribute on the Service Object will be discarded.

If the **Override** attribute is not set, then neither the XML nor the **HardwareId** has precedence. Instead, **PosExplorer** associates the union of all specified devices with the Service Object.

See Also

Tasks

- [Adding Plug and Play Support](#)

Concepts

- [Plug and Play Support](#)
- [POS for .NET Registry Settings](#)

Service Object Samples: Getting Started (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) provides a class tree which implements most of the functionality required by the Unified Point Of Service (UnifiedPOS) specification. In many cases, the Service Object developer only needs to implement the methods and properties required to operate the specific piece of hardware, for which the Service Object is being written.

This section provides a step-by-step guide to creating a simple, but functional sample Service Object for a Magnetic Strip Reader (MSR) device.

In This Section

- [Setting up a Service Object Project](#) Explains how to use Visual Studio to create a class library project with references to the appropriate POS for .NET assemblies.
- [Creating a Basic Service Object Code Template](#) Continues adding to the sample by modifying the code to create the necessary references, attributes, and methods to create a fundamental Service Object template.
- [Adding Plug and Play Support](#) Adds to the sample template by integrating Plug and Play support.
- [Creating a Service Object Sample](#) Describes how the sample code implements the methods needed to compile the sample. The Service Object will now be recognized by POS for .NET applications, but has no functionality.
- [Introducing Service Object Reader Threads](#) Introduces the concept of multithreaded programming in Service Objects. A sample thread helper class is included upon which other multithreaded Service Object samples are built.
- [Creating a Working, Multithreaded Service Object](#) Implements a complete Magnetic Strip Reader (MSR) Service Object. It expands the appropriate methods from the earlier sample so that it returns data to the application. In addition, it uses the thread helper class from the previous section to start and stop a separate read thread.

Related Sections

- [System Configuration](#) Describes how to configure POS for .NET to meet the needs of your installation.
- [Developing a POS Application](#) Explains how to create POS for .NET applications, the ultimate consumers of POS for .NET Service Objects.

Setting up a Service Object Project (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

To begin programming your Service Object, you first need to create a new **Class Library** project and add the appropriate resources to your project. To create your basic project, perform the following steps.

To create a new class library project for your Service Object

1. Open **Visual Studio**. On the **File** menu, choose **New** and then choose **Project** to create a new project.
2. In the **New Project** dialog box, select **Class Library** from the **Visual Studio Installed Templates** list.
3. Choose an appropriate name for your Service Object. Note that **Visual Studio** creates a namespace for your project based on the name that you enter here.

To add references to Point of Service assemblies

1. Be sure that you have installed the **Microsoft Point of Service** product.
2. From the **Solution Explorer** window, right-click the **References** drop-down and then select **Add References**.
3. On the **Browse** tab, locate the **Microsoft.PointOfService** and **Microsoft.PointOfService.ControlBase** assemblies in the **Program Files\Microsoft Point of Service\SDK** directory.
4. Select both assemblies and add them to your reference list.

To add the **PosAssembly** global attribute to your assembly

1. The [PosExplorer](#) requires that Service Object assemblies contain the [PosAssemblyAttribute](#) global attribute.
2. From the Solution Explorer window, open and edit the file **AssemblyInfo.cs**.
3. At the top of the file, add a **using** directive for **Microsoft.PointOfService**, for example:
`using Microsoft.PointOfService;`
4. Insert the global attribute, **PosAssembly**, into the file. This attribute takes a single argument for the name of your organization. For example:
`[assembly: PosAssembly("Your Company Name, Inc")]`

See Also

Concepts

- [Attributes for Identifying Service Objects and Assigning Hardware](#)

Other Resources

- [Service Object Samples: Getting Started](#)

Creating a Basic Service Object Code Template (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The previous section, [Setting up a Service Object Project](#), explained how to create an empty project to begin writing your Service Object. This section continues by adding features to that sample project. The following procedures and the sample in this topic show the steps that you must follow to create a basic Service Object template.

To create a simple class template

1. Add **using** directives for the [Microsoft.PointOfService](#) and [Microsoft.PointOfService.BaseServiceObjects](#) to the top of the source file.
2. Choose the POS for .NET **Base** class your Service Object will be derived from. The **Base** class you choose is based on the type of POS device for which you are developing this Service Object. (See [POS for .NET Class Tree](#))
3. If you are building your class on top of Point of Service **Basic** classes, also add a **using** directive for [Microsoft.PointOfService.BasicServiceObjects](#).
4. Apply a **ServiceObject** attribute to your Service Object class. This includes the following elements:
 - Device Type
 - Service Object name
 - Description of the Service Object
 - Major version
 - Minor version
5. Create a default public parameterless constructor. This is required for [PosExplorer](#) to create an instance of your class by using .NET reflection.

Example

In this sample, notice the additional **using** directives, the **ServiceObject** attribute applied to the Service Object class, the **Base** class used for the Service Object class, and finally the public constructor without arguments.

C#

```
using system;
using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

namespace Samples.ServiceObjects.SOTemplate
{
    [ServiceObject(
        DeviceType.Msr,
        "ServiceObjectTemplate",
        "Bare bones Service Object class",
        1,
        9)]
    public class MyServiceObject : MsrBase
    {
        public MyServiceObject()
        {
        }
    }
}
```

This sample does not compile as is. Its purpose is to demonstrate what elements are necessary for any Service Object class. However, for each **POS for .NET Service Object Base** class, the list of abstract methods which must be implemented is different. The following sections continue to add features to the sample until it becomes a complete, functional Service Object implementation.

See Also

Tasks

- [Setting up a Service Object Project](#)
- [Adding Plug and Play Support](#)

Concepts

- [POS for .NET Class Tree](#)
- [Attributes for Identifying Service Objects and Assigning Hardware](#)

Other Resources

- [Service Object Samples: Getting Started](#)
- [POS for .NET Service Object Architecture](#)

Adding Plug and Play Support (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) includes support for Plug and Play devices. By adding Plug and Play support to your Service Objects, applications can become more simple, reliable, and efficient. Service Objects should support it whenever possible.

Implementing Plug and Play support at the Service Object level is very simple. Once you know the hardware ID of your device, simply add a single attribute to your class, [HardwareIdAttribute](#). The **HardwareId** attribute is used by [PosExplorer](#) to intelligently filter out Service Objects from the list of available devices depending on the state of the device. If the Service Object has a **HardwareId** attribute that refers to an installed Plug and Play device, but that device is not connected, the Service Object will be excluded from the [PosExplorer](#) device list. This list is returned when applications call [GetDevices\(\)](#).

Service Objects may also have more than one **HardwareId** attribute, in which case [PosExplorer](#) associates a union of all specified devices with the Service Object. It is possible to override the **HardwareId** attributes or add to the list of associated hardware on the Service Object without rebuilding the Service Object assembly. For information about overriding or adding the **HardwareId** attribute, see [Plug and Play XML Configuration](#).

Only the application is responsible for catching [DeviceAddedEvent](#) and [DeviceRemovedEvent](#) events and updating its status as appropriate based on the updated device list returned from [PosExplorer](#). The Service Object does not need to detect these events.

To add a **HardwareId** attribute to your Service Object class

1. Determine the range of hardware IDs for the device or devices that your Service Object supports.
2. Add a **HardwareId** attribute before your class definition using the lowest hardware ID used by your device and the highest. Multiple **HardwareId** attributes may be used to identify multiple ranges of hardware IDs.

Example

The following sample adds a **HardwareId** attribute to the basic template shown in the previous section.

```
C#  
  
using System;  
  
using Microsoft.PointOfService;  
using Microsoft.PointOfService.BaseServiceObjects;  
  
namespace SOTemplate  
{  
  
    [HardwareId("HID\\Vid_05e0&Pid_038a",  
               "HID\\Vid_05e0&Pid_038a")]  
  
    [ServiceObject(  
        DeviceType.Msr,  
        "ServiceObjectTemplate",  
        "Bare bones Service Object class",  
        1,  
        9)]  
    public class MyServiceObject : MsrBase  
    {  
        public MyServiceObject()  
        {  
        }  
    }  
}
```

See Also

Tasks

- [Creating a Basic Service Object Code Template](#)
- [Creating a Service Object Sample](#)

Concepts

- [Attributes for Identifying Service Objects and Assigning Hardware](#)
- [Plug and Play XML Configuration](#)

Other Resources

- Service Object Samples: Getting Started
- POS for .NET Service Object Architecture

 Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

The .NET documentation is open source. Provide feedback here.

 [Open a documentation issue](#)

 [Provide product feedback](#)

Creating a Service Object Sample (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Previous topics explained how to create a basic Service Object template with Plug and Play support. This section adds how to create a limited sample with the following new features:

- Necessary abstract methods are implemented so that the sample will compile successfully.
- The Service Object will be recognized by applications using [PosExplorer](#), for example, the **POS for .NET Test Application** included with the SDK.
- Applications may now invoke methods on the Service Object or access properties, although no useful results will be returned.

Requirements

To compile this sample, your project has to have the correct references and global attributes.

Example

C#

```
using System;
using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

namespace Samples.ServiceObjects.MSR
{
    [HardwareId(@"VID\VID_05e0&PID_038a", @"VID\VID_05e0&PID_038a")]

    [ServiceObject(DeviceType.Msr,
        "SampleMsr",
        "Sample Msr Service Object",
        1,
        9)]
    public class SampleMsr : MsrBase
    {
        // String returned from CheckHealth
        private string MyHealthText;

        public SampleMsr()
        {
```

```
// Initialize device capability properties.
Properties.CapIso = true;
Properties.CapTransmitSentinels = true;
Properties.DeviceDescription = "Sample MSR";

// Initialize other class variables.
MyHealthText = "";
}

~SampleMsr()
{
    Dispose(false);
}

// Release any resources managed by this object.
protected override void Dispose(bool disposing)
{
    try
    {
        // Your code here.
    }
    finally
    {
        // Must call base class Dispose.
        base.Dispose(disposing);
    }
}

#region PosCommon overrides
// Returns the result of the last call to CheckHealth().
public override string CheckHealthText
{
    get
    {
        // MsrBasic.VerifyState(mustBeClaimed,
        // mustBeEnabled). This may throw an exception.
        VerifyState(false, false);

        return MyHealthText;
    }
}

public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that device is open, claimed, and enabled.
    VerifyState(true, true);

    // Your code here:
    // check the health of the device and return a
    // descriptive string.

    // Cache result in the CheckHealthText property.
    MyHealthText = "Ok";
    return MyHealthText;
}
```

```

    }

    public override DirectIOData DirectIO(
        int command,
        int data,
        object obj)
    {
        // Verify that device is open.
        VerifyState(false, false);

        return new DirectIOData(data, obj);
    }
#endifregion // PosCommon overrides

#region MsrBasic Overrides
protected override MsrFieldData ParseMsrFieldData(
    byte[] track1Data,
    byte[] track2Data,
    byte[] track3Data,
    byte[] track4Data,
    CardType cardType)
{
    // Your code here:
    // Implement this method to parse track data
    // into fields which will be returned as
    // properties to the application
    // (for example, FirstName,
    // AccountNumber, etc.)
    return new MsrFieldData();
}

protected override MsrTrackData ParseMsrTrackData(
    byte[] track1Data,
    byte[] track2Data,
    byte[] track3Data,
    byte[] track4Data,
    CardType cardType)
{
    // Your code here:
    // Implement this method to convert raw track data.
    return new MsrTrackData();
}
#endifregion
}
}

```

In order to simplify this sample, the code does not implement any globalization features. For example, the value for **Properties.DeviceDescription** would typically be read from a localized strings resource file.

See Also

Tasks

- [Adding Plug and Play Support](#)
- [Introducing Service Object Reader Threads](#)

Concepts

- [POS for .NET Class Tree](#)

Other Resources

- [Service Object Samples: Getting Started](#)

Introducing Service Object Reader Threads (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Most Service Objects need to be able to respond to hardware events asynchronously by starting a separate hardware reader thread. Service Objects are the link between the Point of Service application and the hardware. Therefore, Service Objects must read data from the associated hardware while still being available to the application.

This section demonstrates one way to implement the code necessary for multithreaded Service Objects.

Requirements

To compile this code, the application must include a reference to the `System.Threading` namespace.

The sample below implements a threading helper class which may be used by Service Object implementations, but does not compile or run on its own.

Demonstrates

This sample demonstrates how Service Objects may use threading to support monitoring hardware events asynchronously. The sample code implements a thread helper class which can be used to add basic threading support to a Service Object.

To use the thread helper class provided in this section, you will need to create a class derived from `ServiceObjectThreadHelper`, which is included in the code below, and implement the following methods:

- **ServiceObjectThreadOpen** This method is called from the `OpenThread` method of the thread helper class after initialization has been completed. Implement any hardware-specific initialization code here. This method is virtual. The default implementation simply returns.
- **ServiceObjectThreadClose** This method is called when the thread helper object is terminating its thread or when calling the `Dispose` method and should be used to

release any unmanaged handles or other resources related to the device. This method is virtual. The default implementation simply returns.

- **ServiceObjectProcedure** This method will be invoked once all initialization has taken place and the thread has been started successfully. This method is abstract and must be implemented in the class derived from the thread helper class. The **ServiceObjectProcedure** method takes a single argument, a **ManualEvent** handle. Your thread procedure must exit when this handle is set. This is done by calling **ManualEvent.WaitOne** within a **while** loop. For example:

```
while (true)
{
    // Wait for a hardware event or the thread stop event.

    // Test to see if the thread terminated event is set and
    // exit the thread if so.
    if (ThreadStopEvent.WaitOne(0, false))
    {
        break;
    }

    // The thread is not terminating, so it must be a
    // a hardware event.
}
```

Example

C#

```
using System;
using System.Threading;
using Microsoft.PointOfService;

namespace Samples.ServiceObjects.Advanced
{
    // The following code implements a thread helper class.
    // This class may be used by other Point Of Service
    // samples which may require a separate thread for monitoring
    // hardware.
    public abstract class ServiceObjectThreadHelper : IDisposable
    {
        // The thread object which will wait for data from the POS
        // device.
        private Thread ReadThread;

        // These events signal that the thread is starting or stopping.
    }
}
```

```
private AutoResetEvent ThreadTerminating;
private AutoResetEvent ThreadStarted;

// Keeps track of whether or not a thread should
// be running.
bool ThreadWasStarted;

public ServiceObjectThreadHelper()
{
    // Create events to signal the reader thread.
    ThreadTerminating = new AutoResetEvent(false);
    ThreadStarted = new AutoResetEvent(false);

    ThreadWasStarted = false;

    // You need to handle the ApplicationExit event so
    // that you can properly clean up the thread.
    System.Windows.Forms.Application.ApplicationExit +=
        new EventHandler(Application_ApplicationExit);
}

~ServiceObjectThreadHelper()
{
    Dispose(true);
}

public virtual void ServiceObjectThreadOpen()
{
    return;
}

public virtual void ServiceObjectThreadClose()
{
    return;
}

// This is called when the thread starts successfully and
// will be run on the new thread.
public abstract void ServiceObjectThreadProcedure(
    AutoResetEvent ThreadStopEvent);

private bool IsDisposed = false;
protected virtual void Dispose(bool disposing)
{
    if (!IsDisposed)
    {
        try
        {
            if (disposing == true)
            {
                CloseThread();
            }
        }
        finally
        {

```

```

        IsDisposed = true;
    }
}

public void Dispose()
{
    Dispose(true);

    // This object has been disposed of, so no need for
    // the GC to call the finalization code again.
    GC.SuppressFinalize(this);
}

public void OpenThread()
{
    try
    {
        // Check to see if this object is still valid.
        if (IsDisposed)
        {
            // Throw system exception to indicate that
            // the object has already been disposed.
            throw new ObjectDisposedException(
                "ServiceObjectSampleThread");
        }

        // In case the application has called OpenThread
        // before calling CloseThread, stop any previously
        // started thread.
        SignalThreadClose();

        ServiceObjectThreadOpen();

        // Reset event used to signal the thread to quit.
        ThreadTerminating.Reset();

        // Reset the event that used by the thread to signal
        // that it has started.
        ThreadStarted.Reset();

        // Create the thread object and give it a name. The
        // method used here, ThreadMethod, is a wrapper around
        // the actual thread procedure, which will be run in
        // the threading object provided by the Service
        // Object.
        ReadThread = new Thread(
            new ThreadStart(ThreadMethod));

        // Set the thread background mode.
        ReadThread.IsBackground = false;

        // Finally, attempt to start the thread.
        ReadThread.Start();
    }
}

```

```

        // Wait for the thread to start, or until the time-out
        // is reached.
        if (!ThreadStarted.WaitOne(3000, false))
        {
            // If the time-out was reached before the event
            // was set, then throw an exception.
            throw new PosControlException(
                "Unable to open the device for reading",
                ErrorCode.Failure);
        }

        // The thread has started successfully.
        ThreadWasStarted = true;
    }
    catch (Exception e)
    {
        // If an error occurred, be sure the new thread is
        // stopped.
        CloseThread();

        // Re-throw to let the application handle the
        // failure.
        throw;
    }
}

private void SignalThreadClose()
{
    if(ThreadTerminating != null && ThreadWasStarted)
    {
        // Tell the thread to terminate.
        ThreadTerminating.Set();

        // Give the thread a few seconds to end.
        ThreadStarted.WaitOne(10000, false);

        // Mark the thread as being terminated.
        ThreadWasStarted = false;
    }
}

public void CloseThread()
{
    // Signal the thread that it should stop.
    SignalThreadClose();

    // Call back into the SO for any cleanup.
    ServiceObjectThreadClose();
}

private void Application_ApplicationExit(
    object sender,
    EventArgs e)
{
    SignalThreadClose();
}

```

```
}

// This is the method run on the new thread. First it signals
// the caller indicating that the thread has started
// correctly. Next, it calls the service object's thread
// method which will loop waiting for data or a signal
// to close.
private void ThreadMethod()
{
    try
    {
        // Set the event to indicate that the thread has
        // started successfully.
        ThreadStarted.Set();

        // Call into the thread procedure defined by the
        // Service Object.
        ServiceObjectThreadProcedure(ThreadTerminating);

        // Signal that the thread procedure is exiting.
        ThreadStarted.Set();
    }
    catch (Exception e)
    {
        Logger.Info("ServiceObjectThreadHelper",
                    "ThreadMethod Exception = " + e.ToString());
        throw;
    }
}
}
```

See Also

Tasks

- [Creating a Working, Multithreaded Service Object](#)
- [Adding Plug and Play Support](#)

Other Resources

- [Service Object Samples: Getting Started](#)
- [POS for .NET Service Object Architecture](#)

Creating a Working, Multithreaded Service Object (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The previous sections provided samples and guides to start your project including:

- Creating a simple template.
- Implementing a Service Object class that can be compiled and seen by the Point Of Service sample application via [PosExplorer](#).
- Implementing a thread helper class.

This sample combines all of these steps to create a working, multithreaded MSR Service Object class. This sample does not actually read from any hardware. It simply pushes test data through the system. It illustrates, however, how to add code that is specific to your Service Object.

To customize this code for your Service Object

1. Modify the **PosAssembly** attribute in **AssemblyInfo.cs** so that it contains your organization's name.
2. Be sure that the name of the namespace is appropriate for your organization and Service Object.
3. Modify the **ServiceObject** attribute to contain the type, name, description, and version number of the Service Object that you are creating.
4. Add a **HardwareId** attribute to associate this Service Object with a Plug and Play device, or a range of devices.
5. Include the **ThreadHelper** class presented in [Introducing Service Object Reader Threads](#). You can do this by either pasting the code into your source file, or compiling it as a separate source file in your project. Be sure that the **ThreadHelper** class is in an accessible namespace.
6. Implement the members necessary depending on the **Base** class used by the Service Object and the functionality that you want to support. This sample is a functioning MSR Service Object with very little functionality.

Requirements

To compile this sample, your project has to have the correct references and global attributes. If you have not already created a working POS for .NET Service Object, review the [Service Object Samples: Getting Started](#) sections.

In addition, you will also need to include the code from the previous section, [Introducing Service Object Reader Threads](#), into your project.

Demonstrates

Most Service Objects need to use a second thread to monitor hardware and notify the application of various incoming data events. This sample shows one way to create a multithreaded Service Object. It relies on the **ServiceObjectThreadHelper** class discussed in [Introducing Service Object Reader Threads](#) to do this.

To use the helper class, an application needs to define a new class that implements the **ServiceObjectThreadHelper** interface. This interface includes three methods:

- **ServiceObjectThreadOpen** This method is called during thread initialization and should be used to initialize any hardware-specific resources.
- **ServiceObjectThreadClose** This method is called when the thread is terminated and should be used to release any hardware-specific resources.
- **ServiceObjectThreadProcedure** This method will be called once the thread has been successfully started and should loop waiting on hardware events, and should not exit until the proper **ManualEvent** is triggered.

This code builds on the sample presented in the topic [Creating a Service Object Sample](#) and adds the following features:

- Creates a class derived from **ServiceObjectThreadHelper**.
- Creates an instance of an **MsrThreadingObject** class. The constructor for this class takes a single argument, a reference to the service object.
- Calls methods on the **MsrThreadingObject** object from the Service Object to start and stop the thread helper as appropriate.

Example

C#

```
using System;
using System.Threading;
```

```
using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;
using System.Text;

namespace Samples.ServiceObjects.Advanced.MSR
{
    public class MsrThreadingObject :
        ServiceObjectThreadHelper, IDisposable
    {
        // This is a helper class which will depend on
        // being able to call back into the actual Service
        // Object to pass along data. However, you cannot
        // keep a strong reference to the Service Object,
        // since that will prevent proper disposal, which
        // may create a state in which all hardware resources
        // are not properly released by the SO. Therefore,
        // create a weak reference. From this reference,
        // you can get a temporary strong reference, which
        // you act on and then release.
        WeakReference ServiceObjectReference;

        // The name of the Service Object.
        string ObjectName;

        public MsrThreadingObject(AdvancedSampleMsr so)
        {
            ObjectName = GetType().Name;
            ServiceObjectReference = new WeakReference(so);
        }

        ~MsrThreadingObject()
        {
            Dispose(true);
        }

        private bool IsDisposed = false;
        protected virtual void Dispose(bool disposing)
        {
            if (!IsDisposed)
            {
                IsDisposed = true;
                base.Dispose(disposing);
            }
        }

        public void Dispose()
        {
            Dispose(false);
        }

        #region Methods of ServiceObjectThreadHelper

        // This will be called during initialization.
        public override void ServiceObjectThreadOpen()
        {
```

```
        Logger.Info(ObjectName, "Msr Thread Open");
    }

    // This method will be called during shutdown.
    public override void ServiceObjectThreadClose()
    {
        Logger.Info(ObjectName, "Msr Thread Open");
    }

    public override void ServiceObjectThreadProcedure(
        AutoResetEvent ThreadStopEvent)
    {
        // Convert a C# string into a sample byte array.
        UTF8Encoding encoder = new UTF8Encoding();

        // Convert sample data to a byte array.
        byte[] MsrData = encoder.GetBytes(
            "This is MSR test data");

        Logger.Info(ObjectName, "Msr Thread Procedure Entered");

        while (true)
        {
            // When this method is called by the
            // ServiceObjectThreadHelper, it is obligated to
            // exit when the event ThreadStopEvent has been
            // set.

            // Additionally, this method will also wait for
            // hardware events or for a time-out. That should
            // be done here.

            // This example waits for the event to be set
            // or times out after several seconds.

            if (ThreadStopEvent.WaitOne(2000, false))
            {
                break;
            }
        }

        Logger.Info(ObjectName, "Reader Thread cycling");

        // Try to get a strong reference to the Service
        // Object using the weak reference saved when
        // this helper object was created.
        AdvancedSampleMsr msr =
            ServiceObjectReference.Target
            as AdvancedSampleMsr;

        // If this fails, that means the Service
        // Object has already been disposed of. Exit the
        // thread.
        if (msr == null)
        {
            break;
        }
    }
}
```

```

        }

        // Using the strong reference, you can now make
        // calls back into the Service Object.
        msr.OnCardSwipe(MsrData);
        msr = null;
    }
}

#endregion Methods of ServiceObjectThreadHelper
}

// Implementation of the Service Object class. This class
// implements all the methods needed for an MSR Service
// Object.
//
// A Service Object which supports a Plug and Play device
// should also have a HardwareId attribute here.

[HardwareId(
    @"HID\Vid_05e0&Pid_038a",
    @"HID\Vid_05e0&Pid_038a")]

[ServiceObject(
    DeviceType.Msr,
    "AdvancedSampleMsr",
    "Advanced Sample Msr Service Object",
    1,
    9)]
public class AdvancedSampleMsr : MsrBase
{
    // String returned for various health checks.
    private string MyHealthText;
    private const string PollingStatistic =
        "Polling Interval";

    // Create a class with interface methods called from the
    // threading object.
    MsrThreadingObject ReadThread;
    public AdvancedSampleMsr()
    {
        // DevicePath must be set before Open() is called.
        // In the case of Plug and Play hardware, the POS
        // for .NET Base class will set this value.
        DevicePath = "Sample Msr";

        Properties.CapIso = true;
        Properties.CapTransmitSentinels = true;

        Properties.DeviceDescription =
            "Advanced Sample Msr";

        // Initialize the string to be returned from
        // CheckHealthText().
        MyHealthText = "";
    }
}

```

```

~AdvancedSampleMsr()
{
    // Code added from previous sections to terminate
    // the read thread started by the thread-helper
    // object.
    ReadThread.CloseThread();

    Dispose(false);
}

protected override void Dispose(bool disposing)
{
    try
    {
        if (disposing)
        {
            if (ReadThread != null)
            {
                ReadThread.Dispose();
                ReadThread = null;
            }
        }
    }
    finally
    {
        // Must call base class Dispose.
        base.Dispose(disposing);
    }
}

#region Internal Members
// This is a private method called from the task
// interface when a data event occurs in the reader
// thread.
internal void OnCardSwipe(byte[] CardData)
{
    // Simple sample data.
    UTF8Encoding utf8 = new UTF8Encoding();
    byte[] track1Data = utf8.GetBytes(
        "this is test track 1");
    byte[] track2Data = utf8.GetBytes(
        "this is test track 2");

    // Call GoodRead(), UnreadableCard, or FailedRead
    // from here.
    GoodRead(
        track1Data,
        track2Data,
        null,
        null,
        Microsoft.PointOfService.BaseServiceObjects.CardType.Iso);
}
#endregion Internal Members

```

```

#region PosCommon overrides
// PosCommon.Open.
public override void Open()
{
    // Call base class Open.
    base.Open();

    // Initialize statistic values.

    // Set values for common statistics.
    SetStatisticValue(StatisticManufacturerName,
                      "Microsoft Corporation");
    SetStatisticValue(
        StatisticManufactureDate, "2004-10-23");
    SetStatisticValue(
        StatisticModelName, "Msr Simulator");
    SetStatisticValue(
        StatisticMechanicalRevision, "1.0");
    SetStatisticValue(
        StatisticInterface, "USB");

    // Create a new manufacturer statistic.
    CreateStatistic(
        PollingStatistic,
        false,
        "milliseconds");

    // Set handlers for statistics stored in hardware.
    // Create a class with interface methods called
    // from the threading object.
    ReadThread = new MsrThreadingObject(this);
}

// PosCommon.CheckHealthText.
public override string CheckHealthText
{
    get
    {
        // MsrBasic.VerifyState(mustBeClaimed,
        // mustBeEnabled).
        VerifyState(false, false);
        return MyHealthText;
    }
}

// PosCommon.CheckHealth.
public override string CheckHealth(
    HealthCheckLevel
    level)
{
    // Verify that device is open, claimed, and enabled.
    VerifyState(true, true);

    // Your code here checks the health of the device and
    // returns a descriptive string.
}

```

```
// Cache result in the CheckHealthText property.
MyHealthText = "Ok";
return MyHealthText;
}

// PosCommon.DirectIO.
public override DirectIOData DirectIO(
    int command,
    int data,
    object obj)
{
    return new DirectIOData(data, obj);
}

public override bool DeviceEnabled
{
    get
    {
        return base.DeviceEnabled;
    }
    set
    {
        if (value != base.DeviceEnabled)
        {
            base.DeviceEnabled = value;

            if (value == false)
            {
                // Stop the reader thread when the
                // device is disabled.
                ReadThread.CloseThread();
            }
            else
            {
                try
                {
                    // Enabling the device, start the
                    // reader thread.
                    ReadThread.OpenThread();
                }
                catch (Exception e)
                {
                    base.DeviceEnabled = false;

                    if (e is PosControlException)
                        throw;

                    throw new PosControlException(
                        "Unable to Enable Device",
                        ErrorCode.Failure, e);
                }
            }
        }
    }
}
```

```

        }

#endregion PosCommon overrides.

#region MsrBasic Overrides

// MsrBasic.MsrFieldData
// Once the track data is retrieved, this method is
// called when the application accesses various data
// properties in the MsrBasic class. For example,
// FirstName and AccountNumber.
protected override MsrFieldData ParseMsrFieldData(
    byte[] track1Data,
    byte[] track2Data,
    byte[] track3Data,
    byte[] track4Data,
    CardType cardType)
{
    // MsrFieldData contains the data elements that
    // MsrBasic will return as properties to the
    // application, as they are requested.
    MsrFieldData data = new MsrFieldData();

    // Parse the raw track data and store in fields to
    // be used by the app.
    data.FirstName = "FirstName";
    data.Surname = "LastName";
    data.Title = "Mr.";
    data.AccountNumber = "123412341234";

    return data;
}

// MsrBasic.MsrTrackData.
protected override MsrTrackData ParseMsrTrackData(
    byte[] track1Data,
    byte[] track2Data,
    byte[] track3Data,
    byte[] track4Data,
    CardType cardType)
{
    MsrTrackData data = new MsrTrackData();

    // Modify the track data as appropriate for your SO.
    // Remove the sentinel characters from the track data,
    // for example.
    data.Track1Data = (byte[])track1Data.Clone();
    data.Track2Data = (byte[])track2Data.Clone();

    return data;
}
#endregion MsrBasic overrides
}
}

```

See Also

Tasks

- [Introducing Service Object Reader Threads](#)
- [Adding Plug and Play Support](#)

Other Resources

- [Service Object Samples: Getting Started](#)
- [POS for .NET Service Object Architecture](#)

Developing Service Objects Using Base Classes (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The Unified Point of Service (UnifiedPOS) v1.14 specification defines 36 UnifiedPOS device types. Microsoft Point of Service for .NET (POS for .NET) provides **Base** class for nine of those. This section builds on the basics covered in the [Service Object Samples: Getting Started](#) and provides additional information specific to each device type.

In This Section

- [Scanner Implementation](#) Provides additional details about developing a **ScannerDrawer** Service Object.
- [CashDrawer Implementation](#) Provides additional details about developing a **CashDrawer** Service Object.
- [LineDisplay Implementation](#) Provides additional details about developing a **LineDisplay** Service Object.
- [PinPad Implementation](#) Provides additional details about developing a **PinPad** Service Object.
- [PosKeyboard Implementation](#) Provides additional details about developing a **PosKeyboard** Service Object.
- [Using Impl Methods for Synchronous or Asynchronous Output](#) Describes how these special helper methods allow for a single implementation of an output method that supports both synchronous and asynchronous operations.

Related Sections

- [Service Object Samples: Getting Started](#) Provides a step-by-step guide to creating a functional, multithreaded Service Object.
- [Developing a Custom Service Object](#) Discusses the procedures, issues, and conventions for developing a custom Service Object.

Scanner Implementation (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

A scanner device is used to read bar code data.

The scanner is an exclusive-use device, as follows:

- The application must claim the device before enabling it.
- The application must claim and enable the device before the device begins reading input.

The scanner object follows the general model for event-drive input:

- When input is received from the device, a **DataEvent** event is queued using the Microsoft helper method, **ScannerBase.GoodRead**. If the device receives bad data, the Service Object may also queue an **ErrorEvent** event by calling **ScannerBase.FailedRead**.
- If the **PosCommon.AutoDisable** property is set to **true**, the **ScannerBase** class will set the **PosCommon.EnableDevice** property to **false**. If your Service Object has implemented this method, it will need to disable the device as appropriate.

A queued **DataEvent** event will only be delivered to the application when the property **ScannerBase.DataEventEnabled** is set to **true**.

- The Unified Point Of Service (UnifiedPOS) specification requires that data from the incoming **DataEvent** must be copied into the corresponding properties before being delivered to the application. The method **ScannerBase.PreFireEvent**, which is called just before delivering the **DataEvent** to the application, meets this requirement by calling **ScannerBase.DecodeScanDataLabel** and **ScannerBase.DecodeScanDataType** if the **DecodeData** property is set to **true**. Usually you have to implement these methods in your Service Object.
- Scanned data is placed into the **Scanner.BaseScanData** property. If the application has set the **ScannerBase.DecodeData** property to **true**, then data is decoded into the **ScannerBase.ScanDataLabel** and **ScanDataType** properties.
- Before the **DataEvent** is delivered to the application, the property **ScannerBase.DataEventEnabled** is set to **false**. This prevents further **DataEvents** from being delivered to the application until it has finished processing the current one. The application sets **ScannerBase.DataEventEnabled** to **true** when it is ready to process incoming events.

- The **ScannerBasic.DataCount** property may be read to obtain the total number of queued events.
- All queued events may be deleted by calling the **ScannerBasic.ClearInput** method.

In This Section

- [Data Decoding](#) Describes the code necessary to decode device-specific data.
- [Scanner Events](#) Demonstrates how a scanner Service Object uses POS for .NET queuing to raise events to applications.

Data Decoding (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The [ScannerBase](#) class provides two methods, [DecodeDataLabel](#) and [DecodeScanDataType](#) for decoding incoming date. These methods are called when the properties [ScanDataLabel](#) and [ScanDataType](#), respectively, are accessed. The [ScannerBase](#) class defers data decoding until the application accesses the data properties and the decoded data will be cached for future reads.

The [ScannerBase](#) class implements the [ScannerBase.DecodeData](#) attribute as required by the Unified Point Of Service (UnifiedPOS) specification. If [DecodeData](#) is not set to **true** when the application reads the [ScanDataLabel](#) property, an empty byte array will be returned. Similarly, [ScanDataType](#) returns [BarCodeSymbology.Unknown](#). This functionality is implemented in the [ScannerBase](#) class and is transparent to both the application and the Service Object.

To implement [DecodeScanDataLabel](#)

1. Override the protected, virtual [ScannerBasic](#) member [DecodeScanDataLabel](#).
2. [DecodeScanData](#) takes an argument, *scanData*, which contains the complete data buffer. There is no need to cache any additional data in the Service Object code.
3. [DecodeScanData](#) should process the scanned data to remove header and type information at the start and end of the data buffer. The modified buffer will be returned in a byte array.

To implement [DecodeScanDataType](#)

1. Override the protected, virtual [ScannerBasic](#) member [DecodeScanDataType](#).
2. Like [DecodeScanDataLabel](#), [DecodeScanDataType](#) receives an argument containing the complete scanned buffer.
3. [DecodeScanDataType](#) examines the buffer to find the data type of the scanned data and returns the appropriate [BarCodeSymbology](#) value.

Example

The following code demonstrates a typical method the Service Object developer could implement in order to extract label and data values from a scanned buffer. Note that this code is demonstrative of a particular device. Different Service Objects will require device-specific decoding.

C#

```
// Decode the incoming scanner data, removing header and
// type information.
override protected byte[] DecodeScanDataLabel(
    byte[] scanData)
{
    int i;
    int len = 0;

    // Get length of label data.
    for (i = 5; i < (int)scanData[1]
        && (int)scanData[i] > 31; i++)
    {
        len++;
    }

    // Copy label data into buffer.
    byte[] label = new byte[len];
    len = 0;

    for (i = 5; i < (int)scanData[1]
        && (int)scanData[i] > 31; i++)
    {
        label[len++] = scanData[i];
    }

    return label;
}

// Process the incoming scanner data to find the data type.
override protected BarCodeSymbology DecodeScanDataType(
    byte[] scanData)
{
    int i;

    for (i = 5; i < (int)scanData[1]
        && (int)scanData[i] > 31; i++)
    {
    }

    // last 3 (or 1) bytes indicate symbology.
    if (i + 2 <= (int)ScanData[1])
    {
        return GetSymbology(
            ScanData[i],
            ScanData[i + 1],
            ScanData[i + 2]);
    }
}
```

```

        }
    else
    {
        return GetSymbology(ScanData[i], 0, 0);
    }
}

// This method determines the data type by examining
// the end of the scanned data buffer. Either 1 byte
// or 3 byte is used to determine the type, depending on
// the incoming buffer.
static private BarCodeSymbology GetSymbology(
    byte b1,
    byte b2,
    byte b3)
{
    if (b1 == 0 && b3 == 11)
    {
        // Use all 3 bytes to determine the date type.
        switch (b2)
        {
            case 10:
                return BarCodeSymbology.Code39;
            case 13:
                return BarCodeSymbology.Itf;
            case 14:
                return BarCodeSymbology.Codabar;
            case 24:
                return BarCodeSymbology.Code128;
            case 25:
                return BarCodeSymbology.Code93;
            case 37:
                return BarCodeSymbology.Ean128;
            case 255:
                return BarCodeSymbology.Rss14;
            default:
                break;
        }
    }
    else if (b2 == 0 && b3 == 0)
    {
        // Only use the first byte to determine the data type.
        switch (b1)
        {
            case 13:
                return BarCodeSymbology.UpcA;
            case 22:
                return BarCodeSymbology.EanJan13;
            case 12:
                return BarCodeSymbology.EanJan8;
            default:
                break;
        }
    }
}

```

```
    return BarCodeSymbology.Other;  
}
```

Additional details about how label and type data should be extracted from a scanned data buffer can be found in the UPOS specification.

Compiling the Code

- For additional information about creating and compiling a Service Object project, see [Service Object Samples: Getting Started](#).

See Also

Reference

- [DecodeScanDataLabel\(Byte\[\]\)](#)
- [DecodeScanDataType\(Byte\[\]\)](#)

Other Resources

- [Scanner Implementation](#)

Scanner Events (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Bar code scanners operate asynchronously and therefore have to notify applications when data is available, or the device state has changed. You perform this task by using .NET delegates to raise events to the application.

As discussed in the [Device Input and Events](#) topic, events are queued before they are delivered to the application. The Microsoft Point of Service for .NET (POS for .NET) **Base** classes provide a means for Service Object code to queue events so that their delivery to the application can be deferred until the application can process them. Meanwhile, the Service Object can continue waiting for additional incoming hardware events.

The Scanner device can send four events to the application. For two of these events, **DataEvent** and **ErrorEvent**, the POS for .NET [ScannerBase](#) class provides a protected helper method to simplify the code that is required to raise these events:

Event	Method That Queues the Event
DataEvent	Protected method ScannerBase.GoodRead
ErrorEvent	Protected method ScannerBase.FailedRead

The two other events, **DirectIOEvent** and **StatusUpdateEvent**, must be raised by using members of the lower-level **ScannerBasic** class. For more information, see [Device Input and Events](#).

Because a Scanner device can deliver data to the system at any time, a Scanner Service Object must wait for data asynchronously by starting a separate reader thread. Events should be queued from this thread as data arrives from the device.

To raise events based on device input

1. Start a reader thread to wait for input from the device.
2. Wait for input on the reader thread, most frequently by using Win32 direct functions to read data from the USB bus.
3. After you receive data, verify that the data is valid, for example, that there are enough bytes in the packet for the header and the data type.

4. If the data is not valid, call the **ScannerBase.FailedScan** method to queue an **ErrorEvent** event that will be raised in the application.
5. If the data is valid, call the **ScannerBase.GoodScan** method to queue a **DataEvent** event that will eventually be raised in the application.

Example

As soon as input is received from the device, the Service Object queues the appropriate event. You can do this by writing a method, such as the one in the sample in this topic that would be called from the reader thread of the Service Object.

C#

```
// A Service Object may implement a method such as this one to
// be called from the reader thread of the Service Object.
void OnDataScanned(byte[] data)
{
    // Ignore input if process in the Error state. There is no
    // need to send an ErrorEvent to the application, because it has
    // already been notified by this point.
    if (State == ControlState.Error)
    {
        return;
    }

    // Make sure that the incoming buffer is large enough to contain
    // at least the header and type data.
    if ((int)data[1] < 5)
    {
        // By calling FailedRead, you are queueing an
        // ErrorEvent for eventual delivery to the application.
        FailedScan();
    }
    else
    {
        // The buffer received from the device will be longer
        // than we need. Therefore, trim it down. Allocate space for
        // the number of bytes contained in data[1], plus one
        // more for the first byte in the buffer.
        byte[] b = new byte[(int)data[1] + 1];

        // Copy the data into a new buffer.
        for (int i = 0; i <= (int)data[1]; i++)
        {
            b[i] = data[i];
        }

        // By calling GoodScan, you are queueing a DataEvent
        // which will delivered to the application when it is suitable.
        GoodScan(b);
    }
}
```

```
    }  
}
```

This sample cannot be compiled on its own, but may be inserted into a complete scanner Service Object implementation.

See Also

Tasks

- [Creating a Basic Service Object Code Template](#)

Concepts

- [Base Class DirectIO Method](#)
- [Device Input and Events](#)

Other Resources

- [Service Object Samples: Getting Started](#)

CashDrawer Implementation (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Most point-of-sale applications will use a cash drawer for financial transactions. Service Object developers can use the Microsoft Point of Service for .NET (POS for .NET) class, [CashDrawerBase](#), to easily implement a Unified Point Of Service (UnifiedPOS) compliant [CashDrawer](#) Service Object.

Capabilities

All [CashDrawer](#) Service Objects must support, at a minimum, the ability to open the drawer. This is done by implementing the abstract method,

`CashDrawerBase.OpenDrawerImpl`, in your Service Object class.

The Service Object may also be able to determine if the cash drawer is open or not. If the Service Object does have this capability, it should set the `CapStatus` property to `true`. If `CapStatus` is `true`, then applications may examine the state of the device using the `DrawerOpened` property. If `CapStatus` is not set, then `DrawerOpened` will always be set to `false` and any attempt to set it to `true` will generate an exception.

If `CapStatus` has been set to `true`, the Service Object needs to update the `DrawerOpened` property. You should do this in the Service Object's implementation of the `OpenDrawerImpl` method. A background thread monitoring the state of the device may also set the `DrawerOpened` property.

CashDrawer Events

If the Service Object has set the `CapStatus` property to `true`,

`CashDrawerBasic.DrawerOpened` sends a `StatusUpdateEvent` to the application.

Depending on the cash drawer device and the Service Object implementation, the Service Object may need a separate thread to monitor the state of the hardware and report any changes asynchronously. This would be necessary, for example, if the cash drawer could be opened manually by the operator and the application needs to be notified.

The [CashDrawer](#) Service Object may also send a `DirectIOEvent` to the application. The `DirectIOEvent` is used to send data to the application that is specific to the Service

Object implementation and may therefore be incompatible with some applications. For more information, see [Device Input and Events](#).

Device Sharing

A cash drawer is a shareable device. Multiple applications will be able to open, enable, and access all of its properties and methods. However, once an application has claimed the device, only that application may call `CashDrawerBase.OpenDrawer` or `CashDrawerBase.WaitForDrawerClose`. A `PosException` with `ErrorCode.Claimed` will be thrown if other applications attempt to call these methods.

If more than one application has opened and enabled the device, then each application will receive all events sent by the Service Object.

The code necessary to support this feature is implemented in the POS for .NET `CashDrawerBase` class.

Multiple Cash Drawers

It is possible to have more than one cash drawer attached to the computer and using the same hardware port. In such situations, a `CashDrawer` Service Object may know to which cash drawer it is not specifically connected. If the Service Object can distinguish to which cash drawer device it is connected, it should set the `CapStatusMultiDrawerDetect` property to `true`. The value of this property will influence the behavior of the `DrawerOpened` property and the `WaitForDrawerClose` method.

If `CapStatusMultiDrawerDetect` is set to `false`, then a `DrawerOpened` value of `true` indicates that at least one drawer is open. The application is not able to determine whether any drawer in particular is open.

If `CapStatusMultiDrawerDetect` is set to `false`, the method `WaitForDrawerClose` waits for all open cash drawers to be closed before returning to the application.

See also

- [Developing Service Objects Using Base Classes](#)



Collaborate with us on
GitHub

.NET

.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

LineDisplay Implementation (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

A LineDisplay Service Object is used to display text, bitmaps, and glyphs to a LineDisplay Point of Service device. This section provides information needed to implement LineDisplay Service Objects derived from the [LineDisplayBase](#) class.

In This Section

- [LineDisplay Capabilities](#) Lists the capability attributes and their related methods.
- [LineDisplay Sample](#) Contains a sample framework for a LineDisplay Service Object.

Reference

[Service Object Samples: Getting Started](#)

[Microsoft.PointOfService.BaseServiceObjects](#)

LineDisplay Capabilities (POS for .NET v1.14 SDK Documentation)

Article • 04/07/2023

A LineDisplay Service Object supports, at a minimum, the ability to display characters on the output device. In addition, the device may also support additional features, which the Service Object exposes to the application by setting capability properties and implementing their corresponding methods and properties.

For each additional feature, there is a capability property defined in the [LineDisplayBase](#) class. The capability properties may not be queried until the application has called [Open](#) on the Service Object. Thereafter, the capability properties will indicate which properties and methods may be set and called on the Service Object.

This section lists the features that a LineDisplay Service Object may support. For each feature, there is a capability attribute that must be set by the Service Object as well a set of properties or methods which will be used by the application to access these features. In some cases, the feature is fully supported in [LineDisplayBase](#) and requires no additional code in the Service Object class.

The capability properties are implemented as read-only in order to prevent the application from changing their values. This means, too, that they cannot be set directly by the Service Object. Instead, [LineDisplayBase](#) has a protected property, [Properties](#), which returns a [LineDisplayProperties](#) object. This class provides public equivalents for all capability properties. For example, in order to advertise that it supports blinking, a Service Object would write:

```
Properties.CapBlink = true;
```

And not:

```
CapBlink = true;
```

Marquee-like Scrolling of the Window

The Service Object may support either horizontal or vertical marquees. If horizontal scrolling is supported, the Service Object sets [Properties.CapHMarquee](#) to true. Likewise, if vertical scrolling is supported, [Properties.CapVMarquee](#) is set to true.

Thereafter, applications and Service Objects may use the following to set or get the marquee type:

```
DisplayMarqueeType MarqueeType {get, set; }
```

Inter-Character Wait

A line display device may have the ability to wait for a specified period of time before displaying each character to create a teletype effect. If this feature is supported, the [Properties.CapICharWait](#) property is set to **true**.

Thereafter, applications and Service Objects may use the following to set or get the inter-character wait time:

```
int InterCharacterWait { get; set; }
```

Blinking Text

A line display device may support character-level or device-level blinking at adjustable blink rates. If this feature is supported, the Service Object should set the [Properties.CapBlink](#) property to one of the following [Properties.DisplayBlink](#) enumeration values.

DisplayBlink Value	Corresponding UnifiedPOS Value	Description
None	DISP_CR_NOBLINK	The device does not support blinking.
All	DISP_CR_BLINKALL	The device supports blinking for the entire display.
Each	DISP_CR_BLINKEACH	The device supports blinking for each individual character.

Thereafter, applications and Service Objects may use the following to set or get the blink rate:

```
int BlinkRate {get; set; }
```

Reverse Video

A line display may support character-level or device-level reverse video. If this feature is supported, the Service Object should set the [Properties.CapReverse](#) to a value in the [DisplayReverse](#) enumeration.

DisplayReverse Value	Corresponding UnifiedPOS Value	Description
None	DISP-CR_NONE	Reverse video is not supported.
All	DISP_CR_REVERSEALL	The entire contents of the display are either displayed in reserve video or displayed normally.
Each	DIS_CR_REVERSEEACH	Each character may be individually set to reverse video or normal.

The **CapReverse** property is used by the **DisplayText** method.

Device Descriptors

Descriptors are small indicators with a fixed label, and are typically used to indicate transaction states such as item, total, and change. The Service Object should set **Properties.CapDescriptors** to **true** if descriptors are supported.

Thereafter, applications and Service Objects may use the following to set, get, or clear the Descriptors:

- `int DeviceDescriptors {get; set; }`
- `void ClearDescriptors();`
- `void SetDescriptor(int descriptor, DisplaySetDescriptor attribute);`

Brightness Control

All **LineDisplay** Service Objects support two brightness levels, normal and blank, even if not supported by the physical device. If the device supports additional brightness levels, then **Properties.CapBrightness** should be set to **true**.

Thereafter, applications and Service Objects may use the following to set or get the device brightness:

```
int DeviceBrightness {get; set; }
```

Cursor Attributes

A line display device may support a variety of different cursor types. The **Properties.CapCursorType** property defines which of these types are supported. The **CapCursorType** property is set using the [DisplayCursors](#) enumeration and holds a

bitwise indication of the supported cursor types, which can be any of the following types shown in the table.

CapCursorType enum	UnifiedPOS Value	Description
Blink	DISP_CCT_BLINK	A blinking cursor is supported.
Block	DISP_CCT_BLOCK	Cursor is displayable as a block.
Fixed	DISP_CCT_FIXED	Cursor is always displayed.
HalfBlock	DISP_CCT_HALFBLOCK	Cursor is displayable as a half block.
None	DISP_CCT_NONE	Cursor is not displayable.
Other	DISP_CCT_OTHER	Cursor is displayable, but the form is unknown.
Reverse	DISP_CCT_REVERSE	Cursor is displayable in reverse video.
Underline	DISP_CCT_UNDERLINE	Cursor is displayable as an underline.

Thereafter, applications and Service Objects may use the following to set or get the cursor type:

DisplayCursors `CursorType { get; set; }`

Glyphs

Glyphs are a pixel-level user definition of character cells. If glyphs are supported by the device, then **Properties.CapCustomGlyph** should be set to true.

Thereafter, applications and Service Objects may use the following to set or get the glyph list and settings:

- **RangeOfCharacters** `[] CustomGlyphList { get; set; }`
- `int GlyphHeight { get; }`
- `int GlyphWidth { get; }`
- `void DefineGlyph(int glyphCode, byte[] glyph);`

Screen Modes

A device may support changing the screen mode; that is, the number of rows and columns displayed. If this feature is supported by the device, the Service Object should set **Properties.CapScreenMode** to true.

Thereafter, the application and Service Object may use the following to set or get the screen mode:

- `int ScreenMode { get; set; }`
- `DisplayScreenMode[] ScreenModeList { get; }`

Bitmaps

The Service Object should set the **Properties.CapBitmap** property to **true** if the device supports displaying bitmaps.

The Service Object may want to override the following methods if this capability is supported:

- `void DisplayBitmap(string fileName, int alignmentX, int alignmentY);`
- `void DisplayBitmap(string fileName, int width, int alignmentX, int alignmentY);`

Character Sets

A Service Object should set the **Properties.CapCharacterSet** property with the default character set capability of the line display device. This property can be set to a member of the [CharacterSetCapability](#) enumeration as shown in the following table.

CharacterSetCapability Value	UnifiedPOS Value	Description
Alpha	PTR_CCS_ALPHA	The default character set supports uppercase alphabetic plus numeric, space, minus, and period.
ANSI	N/A	This value is not used for LineDisplay devices.
ASCII	PTR_CCS_ASCII	The default character set supports 0x20 through 0x75.
Kana	PTR_CCS_KANA	The default character set supports partial code page 932, including ASCII characters 0x20 through 0x7F and the Japanese Kana characters 0xA1 through 0xDF, but excluding the Japanese Kanji characters.
Kanji	DISP_CCS_KANJI	The default character set supports code page 932 including the Shift-JIS Kanji characters, Levels 1 and 2.

CharacterSetCapability Value	UnifiedPOS Value	Description
Numeric	N/A	This value is not used for LineDisplay devices.
Unicode	DISP_CCS_UNICODE	The default character set supports UNICODE.
Windows	N/A	This value is not used for LineDisplay devices.

Thereafter, applications and Service Objects may use the following to set or get the character set:

```
int CharacterSet { get; set; }
```

See Also

Tasks

- [LineDisplay Sample](#)

Other Resources

- [LineDisplay Implementation](#)

LineDisplay Sample (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The [LineDisplayBase](#) class is a relatively thin abstraction layer when compared to other Service Object **Base** classes—there is little code needed between the application and the physical device. The [LineDisplay](#) Service Object simply needs to advertise which features the physical device supports and modify its output according to the display properties the application has set.

A [LineDisplay](#) Service Object may also monitor the device and report power or other status changes back to the application using a [StatusUpdateEvent](#). This can be done by using the [Queue](#) methods or, for example, by using the power reporting features in [PosCommon](#). Monitoring the device this way will generally require starting a new thread to wait for hardware events and queue the appropriate [StatusUpdateEvent](#). A [LineDisplay](#) Service Object may also send [DirectIOEvents](#) to the application.

To implement the LineDisplay class and attributes

1. Add `using` directives for the [Microsoft.PointOfService](#) and [Microsoft.PointOfService.BaseServiceObject](#) namespaces.
2. Add the global attribute [PosAssemblyAttribute](#) so that [PosExplorer](#) recognizes this as a Microsoft Point of Service for .NET (POS for .NET) assembly.
3. Create a new class which is derived from [LineDisplayBase](#).
4. Add the class-level attribute [ServiceObjectAttribute](#) to your new class so that [PosExplorer](#) recognizes it as a Service Object.

To implement abstract LineDisplayBase members

1. All [LineDisplay](#) Service Objects must support at least one screen mode. To provide the application with specifics about the supported screen modes, implement the abstract property [LineDisplayScreenModes](#).

2. At a minimum, all **LineDisplay** Service Objects must implement **DisplayData(Cell[])** to display characters on the output device.

Additional Capabilities

Set capability properties in your Service Object to advertise support for the features of your device. This sample demonstrates how to implement the **LineDisplay** blink feature.

To implement the blink feature

1. In the constructor, set the **CapBlink** property to either **DisplayBlink.All** or **DisplayBlink.Each** to indicate which blinking mode this Service Object supports.
2. Set the **CapBlink** property to **true** indicating that the blink rate may be set by the application by calling **BlinkRate**.
3. Take these and other settings into account when implementing **DisplayData**.

Example

C#

```
using System;
using System.Collections.Generic;
using System.Text;

using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

[assembly: PosAssembly("Service Object Contractors, Inc.")]

namespace SOSample.LineDisplay
{
    [ServiceObject(
        DeviceType.LineDisplay,
        "SampleLineDisplay",
        "Sample LineDisplay Service Object",
        1,
        9)]

    public class SampleLineDisplay : LineDisplayBase
    {
        SampleLineDisplay()
        {
            // The CapBlink property is initially set to
            // DisplayBlink.None in LineDisplayBase. This property
            // will be set here to indicate what mode of blinking
        }
    }
}
```

```

// text our Service Object can support.
Properties.CapBlink = DisplayBlink.All;

// Set the CapBlinkRate property to true to indicate
// that this device has the ability to change the
// rate at which it blinks by setting the property
// BlinkRate.
Properties.CapBlinkRate = true;
}

#region Implement Abstract LineDisplayBase Members
// LineDisplayScreenMode must be implemented to
// allow the application to find out which screen modes
// are supported by this device.
protected override LineDisplayScreenMode[]
    LineDisplayScreenModes
{
    get
    {
        LineDisplayScreenMode[] SupportedModes;

        // Create a LineDisplayScreenMode object; this SO
        // has a screen mode 10 columns wide and 2 rows deep.
        LineDisplayScreenMode mode =
            new LineDisplayScreenMode(10, 2, 0, 0);

        // Allocate space for our screen mode array and
        // initialize it to hold our supported screen
        // mode(s).
        SupportedModes =
            new LineDisplayScreenMode[] { mode };

        return SupportedModes;
    }
}

// DisplayData is the method called from the application
// specifying what data should be displayed on the
// device.
protected override void DisplayData(Cell[] cells)
{
    // Your code here:
    // Send the data to your device. Take settings such
    // as blink and blink rate into account here.
    return;
}
#endregion Implement Abstract LineDisplayBase Members

#region Implement Abstract PosCommon Members
private string MyHealthText = "";

// PosCommon.CheckHealthText.
public override string CheckHealthText
{
    get

```

```

    {
        // VerifyState(mustBeClaimed,
        // mustBeEnabled).
        VerifyState(false, false);
        return MyHealthText;
    }
}

// PosCommon.CheckHealth.
public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that the device is open, claimed and enabled.
    VerifyState(true, true);

    // Your code here:
    // check the health of the device and return a
    // descriptive string.

    // Cache result in the CheckHealthText property.
    MyHealthText = "Ok";
    return MyHealthText;
}

// PosCommon.DirectIOData.
public override DirectIOData DirectIO(
    int command,
    int data,
    object obj)
{
    // Verify that the device is open.
    VerifyState(false, false);

    return new DirectIOData(data, obj);
}
#endregion Abstract PosCommon Members
}
}

```

See Also

Other Resources

- [LineDisplay Implementation](#)
- [Developing Service Objects Using Base Classes](#)

PinPad Implementation (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

A **PinPad** device provides a mechanism for customers to perform PIN entry and acts as a cryptographic engine for communicating with an EFT Transaction Host. To perform these tasks, a **PinPad** Service Object may implement one or more PIN Pad Management Systems. A **PinPad** Management System defines the manner in which the **PinPad** performs functions such as PIN Encryption, Message Authentication Code calculations, and Key Updating. Examples of **PinPad** management systems include Master-Session, DUKPT, APACS40, HGEPO2, AS2805, and JDEBIT2, along with many others.

A **PinPad** Service Object must have the following minimum capability:

- Accepts a PIN Entry at its keyboard and provides an Encrypted PIN to the application.

A **PinPad** Service Object may also have the following additional capabilities:

- Computes Message Authentication Codes.
- Performs Key Updating in accordance with the selected PIN Pad Management System.
- Allows use of the PIN Pad Keyboard, Display, and Tone Generator for application usage. If one or more of these features are available, then the application opens and uses the associated POS Keyboard, Line Display, or Tone Indicator Device Objects.

In This Section

- [PinPad Capabilities](#) Outlines the programming model and capabilities for **PinPad** Service Objects.
- [PinPad Sample](#) Provides the **PinPad** sample code.

Reference

- [PinPadBase](#) Provides the Microsoft Point of Service for .NET (POS for .NET) reference for the **PinPadBase** class.

- [Developing a Custom Service Object](#) Describes the POS for .NET Service Object development.

PinPad Capabilities (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

A PIN Pad performs encryption functions under control of a PIN Pad Management System. Some **PinPad** Service Objects support multiple PIN Pad Management Systems and some PIN Pad Management Systems support multiple key sets for different Electronic Funds Transfer (EFT) Transaction Hosts. Thus, for each EFT transaction, the application needs to select the PIN Pad Management System and EFT Transaction Host to be used.

Programming model

Depending on the PIN Pad Management System, one or more EFT transaction parameters need to be provided to the PIN Pad for use in the encryption functions. The application should set the value of ALL EFT Transaction parameter properties to enable easier migration to EFT Transaction Hosts that require a different PIN Pad Management System.

- After opening, claiming, and enabling the PIN Pad Control, an application should use the following general scenario for each EFT Transaction.
- Set the EFT transaction parameters (**AccountNumber**, **Amount**, **MerchantID**, **TerminalID**, **Track1Data**, **Track2Data**, **Track3Data**, **Track4Data**, and **TransactionType** properties) and then call the [BeginEftTransaction\(PinPadSystem, Int32\)](#) method. This will initialize the device to perform the encryption functions for the EFT transaction.

If PIN Entry is **OnFailure**, call the [EnablePinEntry\(\)](#) method. Then set the **DataEventEnabled** property and wait for the **DataEvent** event.

- If Message Authentication Codes are required, use the [ComputeMac\(String\)](#) and [VerifyMac\(String\)](#) methods as needed.
- Call the [EndEftTransaction\(EftTransactionCompletion\)](#) method to notify the device that all operations for the EFT transaction have been completed. This specification supports two models of usage of the display. The **CapDisplay** property indicates one of the following models: - An application has complete control of the text that is to be displayed. For this model, there is an associated **LineDisplay** control that is used by the application to interact with the display. - An application cannot supply the text to be displayed. Instead, it can only select from a list of predefined

messages to be displayed. For this model, there is a set of PIN Pad properties that are used to control the display.

Device sharing

The PIN Pad is an exclusive-use device, therefore:

- The application must claim the device before enabling it.
- The application must claim and enable the device before the device begins reading input, or before calling methods that manipulate the device.

Microsoft Point of Service for .NET (POS for .NET) ~Impl methods

The protected abstract methods that end with the suffix, "Impl" are called from their POS for .NET public counterparts. This allows the **Base** class implementation to perform appropriate status and error checking before and after the ~Impl method is called.

These methods must be implemented in the Service Object code, but the public, nonabstract counterparts should be overridden only in special cases, such as when the Service Object code needs to remove or change the standard validation tests.

POS for .NET events

A PinPad Service Object may send the following events to the application:

- DataEvent
- DirectIOEvent
- StatusUpdateEvent
- ErrorEvent

See Also

Other Resources

- [PinPad Implementation](#)

PinPad Sample (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

This sample demonstrates which methods must be implemented in a **PinPad** Service Object.

To implement a PinPad Service Object framework

1. Add **using** directives for `Microsoft.PointofService`,
`Microsoft.PointOfService.BaseServiceObjects`.
2. Add the global attribute **PosAssembly**.
3. Choose an appropriate namespace name for your project.
4. Create a Service Object class derived from [PinPadBase](#).
5. Add the **ServiceObject** attribute to your Service Object class, using the `DeviceType.PinPad` value as your device type.

Example

C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

using Microsoft.PointOfService.BaseServiceObjects;
using Microsoft.PointOfService;

[assembly: PosAssembly("Service Object Contractors, Inc")]

namespace SOSamples.PinPad
{
    [ServiceObject(
        DeviceType.PinPad,
        "SamplePinPad",
        "Sample PinPad Service Object",
        1,
        9)]
```

```

public class SamplePinPad : PinPadBase
{
    PinPadSystem pinPadSystemSupported = PinPadSystem.Dukpt;

    public SamplePinPad()
    {
    }

    #region Implement Abstract PinPadBase Members

    // These abstract protected methods are called from their
    // public, counterpart methods after error and state
    // validation checks are performed.

    protected override void BeginEftTransactionImpl(
        PinPadSystem pinpadSystem,
        int transactionHost)
    {
        // If pinpadSystem is not supported by this device,
        // throw a PosControlException.
        if (pinpadSystem != pinPadSystemSupported)
        {
            throw new PosControlException(
                "PinPadSystem not supported",
                ErrorCode.Illegal);
        }

        // Your code here. Perform any device-specific
        // initialization, such as computing session keys.
    }

    protected override string ComputeMacImpl(
        string inMsg)
    {
        // Your code here. Depending on the selected PIN Pad
        // Management system, the PinPad Service Object may
        // insert additional fields into the message (inMsg).
        return inMsg;
    }

    protected override void EnablePinEntryImpl()
    {
        // PinPadBase sets PINEntryEnabled if this method
        // succeeds. After that, the Service Object may
        // send a DataEvent to the application.
    }

    protected override void EndEftTransactionImpl(
        EftTransactionCompletion completionCode)
    {
        // Your code here. Perform any device or Service
        // Object cleanup such as computing the next
        // transaction keys.
    }
}

```

```

protected override void UpdateKeyImpl(
    int keyNumber,
    string key)
{
    // Your code here. Update the specified key
    // on your device.
}

protected override void VerifyMacImpl(
    string message)
{
    // Your code here. Verify the MAC value in a message
    // received from the EFT Transaction host.
}

#endregion Implement Abstract PinPadBase Members

#region Implement Abstract PosCommon Members
private string MyHealthText = "";

// PosCommon.CheckHealthText.
public override string CheckHealthText
{
    get
    {
        // VerifyState(mustBeClaimed,
        // mustBeEnabled).
        VerifyState(false, false);
        return MyHealthText;
    }
}

// PosCommon.CheckHealth.
public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that the device is open, claimed, and enabled.
    VerifyState(true, true);

    // Your code here:
    // check the health of the device and return a
    // descriptive string.

    // Cache result in the CheckHealthText property.
    MyHealthText = "Ok";
    return MyHealthText;
}

// PosCommon.DirectIOData.
public override DirectIOData DirectIO(
    int command,
    int data,
    object obj)
{
    // Verify that the device is open.
    VerifyState(false, false);
}

```

```
        return new DirectIOData(data, obj);
    }
#endregion Implement Abstract PosCommon Members
}
}
```

See Also

Other Resources

- [PinPad Implementation](#)

PosKeyboard Implementation (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

A **PosKeyboard** Service Object reads keys from a POS keyboard. A POS keyboard may be an auxiliary keyboard, or it may be a virtual keyboard consisting of some or all of the keys on the system keyboard. In Microsoft Point of Service for .NET (POS for .NET), the POS keyboard **Base** class is [PosKeyboardBase](#).

A **PosKeyboard** Service Object follows the general input device model:

- When input is received from the POS keyboard, a **DataEvent** is queued.
- If the **AutoDisable** property is **true**, then the device automatically disables itself when a **DataEvent** event is queued. This is done automatically by the **PosKeyboardBase** class.
- A queued **DataEvent** event will be delivered to the application when the **DataEventEnabled** property is **true** and other event delivery requirements are met. The **PosKeyboardBase** class will manage event delivery automatically.
- An **ErrorEvent** event is queued if an error while gathering or processing input, and is delivered to the application when **DataEventEnabled** is **true** and other event delivery requirements are met.
- The **DataCount** property, which is maintained by the **PosKeyboardBase** class, may be read to obtain the number of queued events.
- All queued input may be deleted by calling [ClearInput\(\)](#).

The POS keyboard is an exclusive-use device:

- The application must claim the device before enabling it.
- The application must claim and enable the device before the device begins reading input.

This section contains a sample **PosKeyboard** Service Object that generates simulated keystrokes which are sent to the application by using **DataEvents**. This sample depends on the thread-helper object presented in [Introducing Service Object Reader Threads](#). To compile this sample, you need to include the code from that topic.

To write a Service Object

1. Add **using** directives for **Microsoft.PointofService**, **Microsoft.PointOfService.BaseServiceObjects**, and **System.Threading**.

2. Add the global attribute **PosAssembly**.
3. Choose an appropriate namespace name for your project.
4. Create a Service Object class derived from **PosKeyboardBase**.
5. Add the **ServiceObject** attribute to your Service Object class, using the **DeviceType.PosKeyboard** value as your device type.

To add features to the sample keyboard Service Object

1. Create a thread helper class, **KeyboardThreadingObject**, derived from **ServiceObjectThreadHelper** from the Service Object Read Threads section.
2. Implement the **ServiceObjectThreadProcedure** method in the **KeyboardThreadingObject** class. This is the procedure that will be run on a separate thread. In the sample code below, this method simulates keyboard input.
3. This sample class implements a method called **SendKey** and another called **ChangePowerState**. These methods are wrappers around protected members. Because they are protected, they cannot be invoked directly from the threading object.
4. Override the **PosCommon.Open** method to specify capabilities supported by this Service Object and create a new thread-helper object.
5. Override the **PosCommon.DeviceEnable** specifically to Open and Close the thread-helper.
6. Implement the abstract virtual methods from **PosCommon**, providing minimum functionality.

Running the Application

This sample Service Object can be run in conjunction with the test application provided with the POS for .NET Software Development Kit (SDK).

To test the Service Object

1. Start the test application and select **SamplePosKeyboard** from the **Keyboard** drop-down list.

2. Open and Claim the device, and then select the device with the **DeviceEnabled** check box to enable it.
3. Checking the **DataEventEnabled** box will allow the Service Object to send a single simulated key to the application. The **DataEvent** is queued automatically by the **PosKeyboardBase** class when **KeyDown** is called.
4. Selecting the **Automatically enable data events** box allows the Service Object to continue delivering characters, two seconds apart.
5. The Service Object will send simulated key strokes for characters 'a' through 'z'. After that, a **StatusUpdateEvent** event will be sent indicating the device is now offline. This event is sent automatically by the **PosKeyboardBase** class when **Properties.PowerState** is changed.

Example

This sample demonstrates how to develop a simple **PosKeyboard** Service Object. It supports a separate reader thread to send **DataEvents** asynchronously to the application. Once compiled, you can execute the Service Object in conjunction with the test application included with the POS for .NET SDK.

C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;

using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

[assembly: PosAssembly("Service Object Contractors, Inc.")]

namespace SOSamples.Keyboard
{
    [ServiceObject(
        DeviceType.PosKeyboard,
        "SamplePosKeyboard",
        "Sample PosKeyboard Service Object",
        1,
        9)]

    public class SampleKeyboard : PosKeyboardBase
    {
        KeyboardThreadingObject ReadThread = null;

        public SampleKeyboard()
        {
            ReadThread = new KeyboardThreadingObject();
        }

        protected override void OnDataEvent()
        {
            string key = ReadThread.Read();
            if (key != null)
            {
                if (key == "A")
                    Write("B");
                else if (key == "B")
                    Write("C");
                else if (key == "C")
                    Write("D");
                else if (key == "D")
                    Write("E");
                else if (key == "E")
                    Write("F");
                else if (key == "F")
                    Write("G");
                else if (key == "G")
                    Write("H");
                else if (key == "H")
                    Write("I");
                else if (key == "I")
                    Write("J");
                else if (key == "J")
                    Write("K");
                else if (key == "K")
                    Write("L");
                else if (key == "L")
                    Write("M");
                else if (key == "M")
                    Write("N");
                else if (key == "N")
                    Write("O");
                else if (key == "O")
                    Write("P");
                else if (key == "P")
                    Write("Q");
                else if (key == "Q")
                    Write("R");
                else if (key == "R")
                    Write("S");
                else if (key == "S")
                    Write("T");
                else if (key == "T")
                    Write("U");
                else if (key == "U")
                    Write("V");
                else if (key == "V")
                    Write("W");
                else if (key == "W")
                    Write("X");
                else if (key == "X")
                    Write("Y");
                else if (key == "Y")
                    Write("Z");
                else if (key == "Z")
                    Write("A");
            }
        }
    }
}
```

```
{  
    // DevicePath must be set before Open() is called.  
    // In the case of Play and Plug hardware, the  
    // POS for .NET Base class will set this value.  
    DevicePath = "Sample Keyboard";  
  
    // NOTE: You can test the power notification events  
    // sent from this Service Object by selecting the  
    // "Power Notify" check box.  
  
    // Let the application know advanced power  
    // reporting is supported.  
    Properties.CapPowerReporting = PowerReporting.Advanced;  
    Properties.CapKeyUp = false;  
}  
  
~SampleKeyboard()  
{  
    // Code added from previous sections to terminate  
    // the read thread started by the thread-helper  
    // object.  
    if (ReadThread != null)  
    {  
        ReadThread.CloseThread();  
    }  
  
    Dispose(false);  
}  
  
// Expose the protected KeyDown() method so that it can be  
// called from our threading helper.  
public void SendKey(int key)  
{  
    KeyDown(key);  
}  
  
// Expose the protected PowerState property so it can be  
// changed from the threading helper.  
public void ChangePowerState(PowerState state)  
{  
    Properties.PowerState = state;  
}  
  
#region Override Virtual PosCommon Members  
public override void Open()  
{  
    base.Open();  
  
    // Create the reader-thread object.  
    ReadThread = new KeyboardThreadingObject(this);  
}  
  
public override bool DeviceEnabled  
{  
    get
```

```

    {
        return base.DeviceEnabled;
    }
    set
    {
        if (value != base.DeviceEnabled)
        {
            base.DeviceEnabled = value;

            if (value == false)
            {
                // Stop the reader thread when the device
                // is disabled.
                ReadThread.CloseThread();
            }
            else
            {
                try
                {
                    // By enabling the device, start the
                    // reader thread.
                    ReadThread.OpenThread();
                }
                catch (Exception e)
                {
                    base.DeviceEnabled = false;

                    if (e is PosControlException)
                        throw;

                    throw new PosControlException(
                        "Unable to Enable Device",
                        ErrorCode.Failure, e);
                }
            }
        }
    }
}

#endregion Override Virtual PosCommon Members

#region Implement Abstract PosCommon Members
private string MyHealthText = "";

// PosCommon.CheckHealthText.
public override string CheckHealthText
{
    get
    {
        // VerifyState(mustBeClaimed,
        // mustBeEnabled).
        VerifyState(false, false);
        return MyHealthText;
    }
}

```

```

// PosCommon.CheckHealth.
public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that device is open, claimed and enabled.
    VerifyState(true, true);

    // Your code here:
    // Check the health of the device and return a
    // descriptive string.

    // Cache result in the CheckHealthText property.
    MyHealthText = "Ok";
    return MyHealthText;
}

// PosCommon.DirectIOData.
public override DirectIOData DirectIO(
    int command,
    int data,
    object obj)
{
    // Verify that the device is open.
    VerifyState(false, false);

    return new DirectIOData(data, obj);
}
#endifregion Implement Abstract PosCommon Members
}

#region Thread Helper Class
public class KeyboardThreadingObject :
    ServiceObjectThreadHelper, IDisposable
{
    // This is a helper class which will depend on
    // being able to call back into the actual Service
    // Object to pass along data. However, you cannot
    // keep a strong reference to the Service Object,
    // since that may prevent clean disposal, leaving
    // hardware resources unavailable to other processes.
    // Therefore, you create a weak reference. From this
    // reference, you can get a temporary strong reference,
    // which you can act on and then release.
    WeakReference ServiceObjectReference;

    // The name of the Service Object.
    string ObjectName;

    public KeyboardThreadingObject(SampleKeyboard so)
    {
        ObjectName = GetType().Name;
        ServiceObjectReference = new WeakReference(so);
    }

    // This method will be called during initialization.
}

```

```

public override void ServiceObjectThreadOpen()
{
    Logger.Info(ObjectName, "Keyboard Thread Open");
}

// This method will be called during shutdown.
public override void ServiceObjectThreadClose()
{
    Logger.Info(ObjectName, "Keyboard Thread Open");
}

// Your code used to monitor your device for input should
// go here. The implementation below generates simulated
// input as an example.
public override void ServiceObjectThreadProcedure(
    AutoResetEvent ThreadStopEvent)
{
    Logger.Info(ObjectName,
                "Keyboard Thread Procedure Entered");
    int KeyValue = (int)'a';

    while (true)
    {
        // When this method is called by the
        // ServiceObjectThreadHelper, it is obligated to
        // exit when the event ThreadStopEvent has been
        // set.
        if (ThreadStopEvent.WaitOne(2000, false))
        {
            break;
        }

        if (KeyValue <= (int)'z')
        {
            Logger.Info(ObjectName, "Reader Thread cycling");

            // Try to get a strong reference to the Service
            // Object using the weak reference saved when
            // this helper object was created.
            SampleKeyboard Keyboard =
                ServiceObjectReference.Target
                as SampleKeyboard;

            // If this fails, that means the Service Object
            // has already been disposed of - exit the thread.
            if (Keyboard == null)
            {
                break;
            }

            if (Keyboard.DataEventEnabled == true)
            {
                // Call a method implemented in our Keyboard
                // class to queue the key stroke.
                Keyboard.SendKey(KeyValue);
            }
        }
    }
}

```

```

        // Simulate input by moving through the
        // alphabet, sending one character at a time.
        KeyValue++;
        if (KeyValue >= (int)'z')
        {
            // Once you run out of input, simulate a
            // power state change. Setting the SO's
            // PowerState property to
            // PowerState.Offline will cause a
            // StatusUpdateEvent to be sent to the
            // application.
            Keyboard.ChangePowerState(
                PowerState.Offline);

            // Release the strong reference.
            Keyboard = null;

            // There is no more work, so exit the
            // loop.
            break;
        }
    }

    // Release the strong reference.
    Keyboard = null;
}
}

#endregion Thread Helper Class
}

```

Compiling the Code

- This sample requires that the code from the [Introducing Service Object Reader Threads](#) section be included.
- The assemblies **Microsoft.PointOfService** and **Microsoft.PointOfService.BaseServiceObjects** must be referenced.

See Also

Tasks

- [Introducing Service Object Reader Threads](#)

Other Resources

- [Developing Service Objects Using Base Classes](#)
- [Service Object Samples: Getting Started](#)

Using Impl Methods for Synchronous or Asynchronous Output (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Some POS device types support printed or displayed output, for example, the **PosPrinter** or **LineDisplay**. Service Objects written for output-enabled devices should provide support for both asynchronous and synchronous output.

An application that performs an operation asynchronously will set the **AsyncMode** property of the Service Object to **true** before calling the desired output method. The Service Object may throw an exception if it is unable to process the request asynchronously, but this is not ideal since it limits the scope of applications that may use that particular Service Object.

Supporting asynchronous output may, however, require additional and potentially complicated code. This code would manage a queue of incoming requests, maintain a thread to monitor the queue and dispatch requests in the proper order, and raise events back to the application.

The Microsoft Point of Service for .NET (POS for .NET) **Base** classes can manage all of these tasks for the developer. For each output method, the **Base** class has a helper method associated with each output method. These methods, which end with the **Impl** suffix, are written as if they were simply the synchronous implementation of their parent method. The **Base** class transparently manages a queue, dispatches requests to the **Impl** functions at the right time in the right order, and raises the appropriate events back to the application. This same method is called for both synchronous and asynchronous operations.

By deriving from a POS for .NET **Base** class, the Service Object gets asynchronous functionality without implementing additional code.

Base Class Implementation

POS for .NET provides **Base** classes for a subset of POS device types. For each output method in these classes, there is a corresponding method with an **Impl** suffix. The Service Object derived from a **Base** class should override only the **Impl** method and not the corresponding parent method. For example, in a class derived from the POS for .NET

Base class [PosPrinterBase](#), the Service Object would provide an implementation for [PrintNormalImpl](#) and would not override [PrintNormal](#).

The POS for .NET **Base** classes implement all output methods and perform the following tasks in order to support the **Impl** method implemented by the Service Object:

- Validate arguments.
- If the **AsyncMode** property is **false**, the corresponding **Impl** method is called immediately.
- If the **AsyncMode** property is set to **true**:
 1. The request is placed in a queue. This queue is owned by POS for .NET and is managed by its own thread.
 2. Requests on the queue are examined by the queue's thread and the corresponding **Impl** method is called. This call could be delayed if the device is already processing an output request or if a request of higher priority on the queue exists.
- When the **Impl** method returns, the POS for .NET **Base** class code handles all results. For a synchronous call, the method may either return a value or throw an exception. For an asynchronous call, the method may raise [OutputCompleteEvent](#) or [ErrorEvent](#) events back to the application.
- **Impl** methods may also return an object which contains not only error information, but also statistics values.

Service Object **Impl** Methods

Since most processing is handled by the **Base** class code, the Service Object's **Impl** method can be relatively simple.

In many cases, the state of the device at the time of the original call is required by the **Impl** method. In these situations, the **Base** class code saves the current device state along with the output request. The device state is later sent as an argument to the **Impl** method. For example, the definition for [PrintNormalImpl](#) is:

C#

```
protected override PrintResults PrintNormalImpl(  
    PrinterStation station,  
    PrinterState printerState,  
    string data);
```

The argument `printerState` above is specific to the `Impl` method and does not exist in the `PrintNormal(PrinterStation, String)` definition.

The return value of `Impl` functions also differs from their calling methods. For example, note that the `PrintNormalImpl` method returns a class of type `PrintResults`. In addition to `ErrorCode`, `ErrorCodeExtended`, `ErrorLevel`, and `ErrorString`, there are a number of additional properties which will be used by the calling method to update statistic counts.

Example

The following example demonstrates how these methods are implemented in Service Object code.

C#

```
protected override PrintResults PrintNormalImpl(
    PrinterStation station,
    PrinterState printerState,
    string data)
{
    // First, create a PrintResults object to hold return values.
    PrintResults pr = new PrintResults();

    // Now print, depending on the station.
    if (station == PrinterStation.Receipt)
    {
        // Your code goes here.

        // Update statistics to be returned to the caller.
        pr.ReceiptLinePrintedCount = 1;
        pr.ReceiptCharacterPrintedCount = data.Length;
    }
    else if (station == PrinterStation.Slip)
    {
        // Your code goes here.

        // Update statistics to be returned to the caller.
        pr.SlipLinePrintedCount = 1;
        pr.SlipCharacterPrintedCount = data.Length;
    }

    return pr;
}
```

See Also

Tasks

- [Asynchronous Output Sample](#)

Reference

- [PosPrinterBase](#)

Concepts

- [Device Output Models](#)

Other Resources

- [Developing a Custom Service Object](#)

Device Input and Events (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

All POS devices have the ability to generate events or change state independent of the application. For example, if an operator unplugs a **PinPad** device, the application has no direct way of detecting this change since it is not a state change requested by the application. A Service Object must have some way of alerting the application to these state changes.

Multithreading

Since having the application continuously poll the Service Object for its current state would be far too expensive, another solution is needed. Typically, the solution is to create a background thread to monitor the device.

As other examples have demonstrated, creating a reader thread is always necessary for input devices such scanners or magnetic strip readers. For output devices such as line displays and printers, however, a second thread is often necessary to watch for state changes, such as losing power or going offline, and then to send a **StatusUpdateEvent** event to the application.

In this way, the Service Object can be responsive to requests from the application while asynchronously monitoring the hardware.

Defining Events

Events are the mechanism by which the Service Object notifies the application of a state change in the device, or the arrival of new data.

In general terms, an event is a notification between one thread or process and another that something has occurred. More specifically, Microsoft Point of Service for .NET (POS for .NET) uses the .NET delegates feature to deliver to the application.

The Unified Point Of Service (UnifiedPOS) specification defines a set of five events: **DataEvent**, **DirectIOEvent**, **ErrorEvent**, **OutputCompleteEvent**, and **StatusUpdateEvent**. Each Service Object may be permitted to only support a subset of these. The exact contents of the data also depends on the Service Object type.

Event Queues

When you create a Service Object class derived from one of the POS for .NET **Base** classes, events are not sent directly from the Service Object to the application. Instead, events are placed into a queue managed by the **Base** class. Since there are conditions that must be met before events may be delivered to an application, code in the **Base** class dispatches events only when it is appropriate to do so. The Service Object does not need to be aware of the queue or of the requirements that must be met before an event can be fired. This greatly eases the burden on the Service Object developer.

The event queue operates asynchronously by using its own thread. This means that the Service Object does not wait for the actual delivery of the event.

Adding Events to the Queue

The POS for .NET **Base** classes provide a number of ways to add an event to the queue depending on the Service Object and the event type.

Many **Base** classes have helper methods to simplify the queuing of certain events; most commonly, **DataEvent** events. For instance, the method **MsrBase.GoodRead** can be used to queue a **DataEvent** event after a successful card read. Likewise, **PosKeyboard.KeyDown** queues a **DataEvent** indicating that a key has been pressed.

Events may also be queued automatically by the **Base** class when a certain state is changed. For example, if a Service Object has set its **Properties.CapPowerReporting** property, then a **StatusUpdateEvent** indicating a power change can be sent simply by setting the **Properties.PowerState** property in the Service Object.

Finally, if needed a Service Object may specifically queue an event using any of the **QueueEvent** overrides. This may be used most often for sending a **DirectIOEvent**. Since **DirectIOEvent** events are vendor-specific and device-specific, no generic mechanism can be used to queue them.

Synchronous Input

Although most device input is read asynchronously by the Service Object and then dispatched to the application in the form of events, there are instances, however, where the application may request data from the Service Object and not return until the data is ready or a time-out has been reached. For more information about event-driven input, see [Event Management](#).

See Also

Tasks

- [Event Handler Sample](#)

Concepts

- [Event Management](#)

Other Resources

- [Developing Service Objects Using Base Classes](#)

Device Output Models (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The Unified Point Of Service (UnifiedPOS) output model consists of two output types; synchronous and asynchronous. A POS device type may support one or both types, or neither type.

Synchronous Output

When an application uses a device type-specific synchronous method to write output, the write operation takes place on the same thread that called the method. The Service Object may not return until the write operation has either been completed or failed.

Using synchronous output is simple, but can potentially impact application performance if the output cannot be completed relatively quickly. Service Object developers should take this into account.

Asynchronous Output

Certain POS device types support asynchronous output. In the asynchronous output model, the application calls the Service Object to request that data be output to the device. Unlike the synchronous model, however, the Service Object must not wait for the write operation to complete; instead it should return control to the application as soon as possible. When a Service Object receives a request from the application, it should do the following:

- If the physical device is not able to receive data, the Service Object should buffer it in memory until the device is ready.
- Set the **OutputId** property to an identifier for this request, to be used during future events that are sent to the application.
- Return as soon as possible.

The Service Object must then wait for the device to complete the request. Typically, this is done with a separate thread, managed by the Service Object, which monitors the hardware. Once the request is completed successfully, an **OutputCompleteEvent** event, with **OutputEventArgs.OutputId** set to the previously specified identifier, is queued for delivery to the application.

Service Object Managed Queue

The POS for .NET class library offers support for asynchronous output, which is sufficient for nearly all Service Object scenarios.

There are, however, some scenarios where Service Object developers may need to implement their own asynchronous output handling. The primary scenario is to support devices that support hardware-based print queues. In this case, the Service Object sets **UseExternalPrintQueue** to **true**, overrides the **PreQueuePrintData** method, and implements their own queue mechanism.

When **UseExternalPrintQueue** is set to **true**, the **Base** class no longer adds the print requests to its internal asynchronous queue, so it is up to the Service Object developers to queue data in any way they require. This is often done by using the device's hardware print queuing features. The **Base** class still prevalidates the same print requests but does not do any additional processing.

In these cases, the Service Object will be responsible for the following:

- Implementing its own queuing logic.
- Sending **StatusUpdateEvents** for successful operations.
- Sending **ErrorEvents** for failed asynchronous operations and handling the retry.
- Updating the state property.
- All other asynchronous operations defined in the UnifiedPOS specification.

See Also

Tasks

- [Event Handler Sample](#)
- [Asynchronous Output Sample](#)
- [Introducing Service Object Reader Threads](#)

Other Resources

- [Developing Service Objects Using Base Classes](#)

Asynchronous Output Sample (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Microsoft Point of Service for .NET (POS for .NET) supports asynchronous output in compliance with the Unified Point Of Service (UnifiedPOS) specification. In the asynchronous output model, the Service Object must queue output requests so that it can return control to the application as quickly as possible. A second thread must then dispatch output to the device and notify applications when the request has been fulfilled, either with an **OutputCompleteEvent** or an **ErrorEvent** event.

The POS for .NET class library handles most of these functions for the Service Object developer so that there is little, if any, difference between an asynchronous output device and a synchronous output only device.

To create the project

1. Create a Visual Studio class library project.
2. Add the sample code below to your project.
3. Add references to the **Microsoft.PointOfService** assemblies.
4. Compile and copy the Service Object to one of the directories in your Service Object assembly load path.

To use the application sample with the Service Object

- This Service Object can be used with the application sample presented in [Event Handler Sample](#).

Example

To output to a **PosPrinter** device, an application will most commonly use the [PrintNormal\(PrinterStation, String\)](#) method. Notice that the **PosPrinter** Service Object code below does not provide an implementation for this method. Instead, [PrintNormalImpl\(PrinterStation, PrinterState, String\)](#) is implemented. This method is

called by the POS for .NET library for both synchronous and asynchronous output requests.

When an application calls an output method, such as **PrintNormal**, the POS for .NET implementation checks the value of the [AsyncMode](#) property. If this value is **false**, then the POS for .NET library sends the request to **PrintNormalImpl** immediately and waits for it to return. If the value is **true**, however, then the POS for .NET implementation of **PrintNormal** adds the request to an internally managed queue. While there are items in the queue, a POS for .NET managed thread will send each request, in first-in-first-out (FIFO) order, to the device by calling **PrintNormalImpl**. When **PrintNormalImpl** returns, the library implementation will raise an [OutputCompleteEvent](#) in the application. In short, the same Service Object code can support both synchronous and asynchronous output without ever needing to know which output mode is being used.

C#

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

[assembly: PosAssembly("Service Object Contractors, Inc.")]

namespace SOSamples.AsyncOutput
{
    [ServiceObject(
        DeviceType.PosPrinter,
        "AsyncOutputPrinter",
        "Sample Async Printer",
        1,
        9)]

    public class AsyncOutputSimulator : PosPrinterBase
    {
        public AsyncOutputSimulator()
        {
            DevicePath = "Sample Async Printer";

            // Indicate that the Service Object supports
            // the receipt printer.
            Properties.CapRecPresent = true;
        }

        // Note that this method will be called by the POS for .NET
        // library code, regardless of whether the print request
        // is synchronous or asynchronous. The print request
        // queue is managed completely by POS for .NET so the
        // Service Object should simply write data to the device
    }
}
```

```

// here.

protected override PrintResults PrintNormalImpl(
    PrinterStation station,
    PrinterState printerState,
    string data)
{
    // Your code to print to the actual hardware would go
    // here.

    // For demonstration, however, the code simulates
    // that fulfilling this print request took 4 seconds.
    Thread.Sleep(4000);

    PrintResults results = new PrintResults();
    return results;
}

// This method must be implemented by the Service
// Object, and should validate the data to be printed,
// including any escape sequences. This method should throw
// a PosControlException to indicate failure.
protected override void ValidateDataImpl(
    PrinterStation station,
    string data)
{
    // Insert your validation code here.
    return;
}

#region Implement Abstract PosCommon Members
private string MyHealthText = "";

// PosCommon.CheckHealthText.
public override string CheckHealthText
{
    get
    {
        // VerifyState(mustBeClaimed,
        // mustBeEnabled).
        VerifyState(false, false);
        return MyHealthText;
    }
}

// PosCommon.CheckHealth.
public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that device is open, claimed, and enabled.
    VerifyState(true, true);

    // Insert your code here:
    // check the health of the device and return a
    // descriptive string.
}

```

```

        // Cache result in the CheckHealthText property.
        MyHealthText = "Ok";
        return MyHealthText;
    }

    // PosCommon.DirectIOData.
    public override DirectIOData DirectIO(
        int command,
        int data,
        object obj)
    {
        // Verify that the device is open.
        VerifyState(false, false);

        return new DirectIOData(data, obj);
    }
    #endregion Implement Abstract PosCommon Members
}
}

```

The application code in the [Event Handler Sample](#) can be compiled and run with this Service Object.

See Also

Tasks

- [Event Handler Sample](#)

Concepts

- [Event Management](#)
- [Device Output Models](#)

Other Resources

- [Developing a Custom Service Object](#)

Statistics Sample (POS for .NET v1.14 SDK Documentation)

Article • 05/25/2023

Microsoft Point of Service for .NET (POS for .NET) Service Objects continually track and update a number of device statistics for each connected device. The statistic tracking is accomplished using the [DeviceStatistics](#) class, which contains methods for resetting, retrieving, and updating statistics, as well as helper methods for creating statistics, incrementing statistics, and loading or saving statistics to or from XML file storage.

Statistics reporting in POS for .NET supports statistics stored in either hardware or software. Software-based statistics are stored in an XML statistics file, \Program Files\Common Files\Microsoft Shared\Point of Service\Statistics\PosDeviceStatistics.xml, and updated at a globally defined flush interval which is determined at the time of system installation and setup. By default, this flush interval is 1 second. When the application claims a device, POS for .NET loads the appropriate statistics from the XML file.

Device specific statistics management is supported by the **Base** or **Basic** device class interface. Service Objects inheriting from a **Base** or **Basic** class are typically only required to call **IncrementStatistic**, which increments the value of a specified statistic, or **SetStatistic**, which sets one statistic to a specified value. All other statistics implementation is provided by the **Base** or **Basic** class interface.

The key members of the [DeviceStatistics](#) class are detailed below:

- **CapStatisticsReporting** identifies the statistics reporting capabilities of the device. When **CapStatisticsReporting** is set to **false**, no statistical data regarding the device is available. If **CapStatisticsReporting** is **true**, some statistical data is available, and the device may begin accumulating various statistics regarding usage. The information accumulated and reported is device-specific, and is retrieved using the **RetrieveStatistics** method.
- **CapUpdateStatistics** defines whether the gathered statistics can be reset or updated by an application employing the **UpdateStatistics** and **ResetStatistics** methods. This property is only valid if **CapStatisticsReporting** is **true**. If **CapStatisticsReporting** is **false**, the **CapUpdateStatistics** property is always **false**.
- **ResetStatistics** can only be called if both **CapStatisticsReporting** and **CapUpdateStatistics** are **true**. This method resets one, some, or all of the resettable device statistics to zero. It resets the defined resettable statistics in a

device to zero. All the requested statistics must be successfully reset in order for this method to complete successfully. Otherwise, an error is thrown with the Extended **ErrorCode** method. This method is always executed synchronously.

- **UpdateStatistics** sets one, some, or all of the resettable device statistics to the specified values. All the requested statistics must be successfully updated in order for this method to complete successfully. If some cannot be updated for any reason, an **ErrorCode** event is returned. Both **CapStatisticsReporting** and **CapUpdateStatistics** must be set to **true** to call this method. The **UpdateStatistics** method is always executed synchronously.
- **RetrieveStatistics** retrieves the requested statistics from a device. **CapStatisticsReporting** must be **true** in order to successfully use this method. All calls to **RetrieveStatistics** will return the following XML as a minimum.

XML

```
<?xml version='1.0'?>
<UPOSStat version="1.10.0"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xmlns="http://www.nrf-arts.org/IXRetail/namespace/"
xsi:schemaLocation="http://www.nrf-arts.org/IXRetail/namespace/
UPOSStat.xsd">
    <Event>
        <Parameter>
            <Name>RequestedStatistic</Name>
            <Value>1234</Value>
        </Parameter>
    </Event>
    <Equipment>
        <UnifiedPOSVersion>1.10</UnifiedPOSVersion>
        <DeviceCategory UPOS="CashDrawer"/>
        <ManufacturerName>Cashdrawers R Us</ManufacturerName>
        <ModelName>CD-123</ModelName>
        <SerialNumber>12345</SerialNumber>
        <FirmwareRevision>1.0 Rev. B</FirmwareRevision>
        <Interface>RS232</Interface>
        <InstallationDate>2000-03-01</InstallationDate>
    </Equipment>
</UPOSStat>
```

If the application requests a statistic name that the device does not support, the **<Parameter>** entry will be returned with an empty **<Value>**. For example:

XML

```
<Parameter>
    <Name>RequestedStatistic</Name>
```

```
<Value></Value>  
</Parameter>
```

All statistics that the device collects that are manufacturer specific and not defined in the schema will be returned in a `<ManufacturerSpecific>` tag instead of a `<Parameter>` tag. For example:

XML

```
<ManufacturerSpecific>  
  <Name>TheAnswer</Name>  
  <Value>42</Value>  
</ManufacturerSpecific>
```

When an application requests all statistics from the device, the device returns a `<Parameter>` entry for every defined statistic in the device category. The device category is defined by the version of the XML schema specified by the **version** attribute in the `<UPOSStat>` tag. If the device does not record any statistics, the `<Value>` tag will be empty.

POS for .NET uses handlers to perform the reading and writing of statistics in a manner similar to the way events are handled. When one of these statistic handlers is created, it is assigned to a particular device statistic. When this statistic is read or updated, the handler calls a delegate that is able to read from or write to the device as necessary.

Statistic handlers use different types of delegates to perform different statistics operations. **GetStatistic** delegates are used to retrieve statistic values, while **SetStatistic** delegates assign statistic values. **GetStatistics** delegates take the name of the statistic to be retrieved as a parameter and return a string representing the value of that statistic. **SetStatistic** delegates take the name of the statistic to be changed and the value which is to be assigned to that statistic as parameters. **SetStatistic** delegates do not return any values.

Statistics handlers may be assigned to any of the standard statistics defined by the Unified Point Of Service (UnifiedPOS) specification, but custom handlers for any statistics that are created using **CreateStatistic** by Service Object code are also supported.

Example

The following code example demonstrates how to enable, handle, and retrieve statistics data from an MSR device.

C#

```
// File FORM1.CS
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Xml;
using System.IO;

using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

namespace Statistics
{
    public partial class SampleStatistics : Form
    {
        // Indicates whether or not the Service Object has
        // been started.
        bool ServiceObjectStarted = false;

        // The Service Object.
        PosCommon so = null;

        public SampleStatistics()
        {
            InitializeComponent();

            // Disable the buttons until the SO is loaded successfully.
            UpdateControls();
        }

        public void UpdateControls()
        {
            btnGenerateStatistics.Enabled = ServiceObjectStarted;
            btnRetrieveStatistics.Enabled = ServiceObjectStarted ;
            txtStatisticName.Enabled = ServiceObjectStarted;

            // The statistic name text box is disabled until the
            // Service Object is loaded.
            if (ServiceObjectStarted)
            {
                btnStartSO.Text = "Close SO";
            }
            else
            {
                txtStatisticName.Clear();
                txtRetrievedStatistics.Clear();
                btnStartSO.Text = "Start SO";
            }
        }
}
```

```

// The retrieve one statistic button is disabled until
// the user has entered a statistic name.
if (txtStatisticName.TextLength > 0)
{
    btnRetrieveStatistic.Enabled = true;
}
else
{
    btnRetrieveStatistic.Enabled = false;
}
}

private void StartServiceObject(object sender, EventArgs e)
{
    PosExplorer explorer = new PosExplorer(this);
    string S0Name = "SampleStatistics";

    if (ServiceObjectStarted)
    {
        so.DeviceEnabled = false;
        so.Close();
        so = null;
        ServiceObjectStarted = false;
        UpdateControls();
    }
    else
    {
        foreach (DeviceInfo d in explorer.GetDevices())
        {
            if (d.ServiceObjectName == S0Name)
            {
                try
                {
                    // Standard Service Object startup.
                    so = explorer.CreateInstance(d)
                        as PosCommon;

                    so.Open();
                    so.Claim(0);
                    so.DeviceEnabled = true;

                    // Application-specific setup.
                    ServiceObjectStarted = true;
                    UpdateControls();
                }
                catch
                {
                    // Something went wrong starting the SO
                    MessageBox.Show("The Service Object '" +
                        S0Name + "' failed to load",
                        "Service Object Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Exclamation);
                    return;
                }
            }
        }
    }
}

```

```
                break;
            }
        }

        if (so == null)
        {
            // No Service Object with the
            // specified name could be found.
            MessageBox.Show("The Service Object '" +
                SOName + "' could not be found",
                "Service Object Error",
                MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation);
        }
    }
}

private void GenerateStatistics(object sender, EventArgs e)
{
    // In this example case, you use this
    // property to tell the Service Object to change statistic
    // values.
    so.DeviceEnabled = true;

    // Report status.
    txtRetrievedStatistics.Text = "DeviceEnabled called to" +
        " to modify statistic values.";
}

private void RetrieveStatistics(object sender, EventArgs e)
{
    string statistics;
    bool IsXml = true;

    try
    {
        statistics = so.RetrieveStatistics();
    }
    catch
    {
        statistics = "No statistics found";
        IsXml = false;
    }

    DisplayStatistics(statistics, IsXml);
}

private void RetrieveOneStatistic(object sender, EventArgs e)
{
    string statistics;
    bool IsXml = true;

    try
    {
```

```

        statistics = so.RetrieveStatistic(
            txtStatisticName.Text);
    }
    catch
    {
        statistics = "Statistic not found: " +
            txtStatisticName.Text;

        IsXml = false;
    }

    DisplayStatistics(statistics, IsXml);
    txtStatisticName.Clear();
    btnRetrieveStatistic.Enabled = false;
}

private void StatisticSizeChanged(object sender, EventArgs e)
{
    if (txtStatisticName.TextLength > 0)
    {
        btnRetrieveStatistic.Enabled = true;
    }
    else
    {
        btnRetrieveStatistic.Enabled = false;
    }
}

// When retrieving statistics, POS for .NET returns a block
// of XML (as specified in the UPOS specification). This
// method will make it look readable with white space
// and indenting and then display it in the text box.
private void DisplayStatistics(string inputString, bool isXml)
{
    string s = null;

    // In case of an exception, you do not have an XML
    // string, so just display the error description. Otherwise,
    // load the XML string into an XmlDocument object and
    // make it look readable.
    if (!isXml)
    {
        s = inputString;
    }
    if(isXml)
    {
        // Create new XML document and load the statistics
        // XML string.
        XmlDocument doc = new XmlDocument();
        doc.LoadXml(inputString);

        // Create a XmlTextWriter using a MemoryStream and
        // tell it to indent the XML output (so that it is
        // readable.)
        MemoryStream m = new MemoryStream();

```

```

        XmlTextWriter writer = new XmlTextWriter(m, null);
        writer.Formatting = Formatting.Indented;
        writer.Indentation = 4;

        // Save the document to the memory stream using the
        // XmlWriter.
        doc.Save(writer);

        // The stream will be encoded as UTF8, so convert the
        // buffer into a string.
        UTF8Encoding u = new UTF8Encoding();
        s = u.GetString(m.GetBuffer());
    }

    // Write the string to the text box.
    txtRetrievedStatistics.Text = s;
}
}
}

```

C#

```

// File FORM1.DESIGNER.CS
namespace Statistics
{
    partial class SampleStatistics
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()

```

```
    {
        this.btnAddSO = new System.Windows.Forms.Button();
        this.btnGenerateStatistics = new System.Windows.Forms.Button();
        this.btnRetrieveStatistics = new System.Windows.Forms.Button();
        this.txtStatisticName = new System.Windows.Forms.TextBox();
        this.txtRetrievedStatistics = new
System.Windows.Forms.TextBox();
        this.btnRetrieveStatistic = new System.Windows.Forms.Button();
        this.groupBox1 = new System.Windows.Forms.GroupBox();
        this.groupBox1.SuspendLayout();
        this.SuspendLayout();
        // 
        // btnStartSO
        // 
        this.btnAddSO.Location = new System.Drawing.Point(45, 35);
        this.btnAddSO.Name = "btnAddSO";
        this.btnAddSO.Size = new System.Drawing.Size(133, 23);
        this.btnAddSO.TabIndex = 0;
        this.btnAddSO.Text = "Start SO";
        this.btnAddSO.UseVisualStyleBackColor = true;
        this.btnAddSO.Click += new
System.EventHandler(this.StartServiceObject);
        // 
        // btnGenerateStatistics
        // 
        this.btnGenerateStatistics.Location = new
System.Drawing.Point(45, 75);
        this.btnGenerateStatistics.Name = "btnGenerateStatistics";
        this.btnGenerateStatistics.Size = new System.Drawing.Size(133,
23);
        this.btnGenerateStatistics.TabIndex = 1;
        this.btnGenerateStatistics.Text = "GenerateStatistics";
        this.btnGenerateStatistics.UseVisualStyleBackColor = true;
        this.btnGenerateStatistics.Click += new
System.EventHandler(this.GenerateStatistics);
        // 
        // btnRetrieveStatistics
        // 
        this.btnRetrieveStatistics.Location = new
System.Drawing.Point(45, 117);
        this.btnRetrieveStatistics.Name = "btnRetrieveStatistics";
        this.btnRetrieveStatistics.Size = new System.Drawing.Size(133,
23);
        this.btnRetrieveStatistics.TabIndex = 2;
        this.btnRetrieveStatistics.Text = "Retrieve Statistics";
        this.btnRetrieveStatistics.UseVisualStyleBackColor = true;
        this.btnRetrieveStatistics.Click += new
System.EventHandler(this.RetrieveStatistics);
        // 
        // txtStatisticName
        // 
        this.txtStatisticName.Location = new System.Drawing.Point(16,
68);
        this.txtStatisticName.Name = "txtStatisticName";
        this.txtStatisticName.Size = new System.Drawing.Size(205, 20);
    }
```

```
    this.txtStatisticName.TabIndex = 4;
    this.txtStatisticName.TextChanged += new
System.EventHandler(this.StatisticSizeChanged);
    //
// txtRetrievedStatistics
//
    this.txtRetrievedStatistics.BackColor =
System.Drawing.Color.White;
    this.txtRetrievedStatistics.Location = new
System.Drawing.Point(45, 157);
    this.txtRetrievedStatistics.Multiline = true;
    this.txtRetrievedStatistics.Name = "txtRetrievedStatistics";
    this.txtRetrievedStatistics.ReadOnly = true;
    this.txtRetrievedStatistics.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
    this.txtRetrievedStatistics.Size = new System.Drawing.Size(476,
247);
    this.txtRetrievedStatistics.TabIndex = 5;
//
// btnRetrieveStatistic
//
    this.btnRetrieveStatistic.Location = new
System.Drawing.Point(16, 30);
    this.btnRetrieveStatistic.Name = "btnRetrieveStatistic";
    this.btnRetrieveStatistic.Size = new System.Drawing.Size(133,
23);
    this.btnRetrieveStatistic.TabIndex = 6;
    this.btnRetrieveStatistic.Text = "Retrieve Statistic";
    this.btnRetrieveStatistic.UseVisualStyleBackColor = true;
    this.btnRetrieveStatistic.Click += new
System.EventHandler(this.RetrieveOneStatistic);
//
// groupBox1
//
    this.groupBox1.Controls.Add(this.btnRetrieveStatistic);
    this.groupBox1.Controls.Add(this.txtStatisticName);
    this.groupBox1.Location = new System.Drawing.Point(219, 35);
    this.groupBox1.Name = "groupBox1";
    this.groupBox1.Size = new System.Drawing.Size(302, 105);
    this.groupBox1.TabIndex = 7;
    this.groupBox1.TabStop = false;
    this.groupBox1.Text = "Single Statistic";
//
// SampleStatistics
//
    this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.BackColor = System.Drawing.SystemColors.MenuBar;
    this.ClientSize = new System.Drawing.Size(565, 439);
    this.Controls.Add(this.groupBox1);
    this.Controls.Add(this.txtRetrievedStatistics);
    this.Controls.Add(this.btnRetrieveStatistics);
    this.Controls.Add(this.btnGenerateStatistics);
    this.Controls.Add(this.btnStartSO);
    this.Name = "SampleStatistics";
```

```

        this.Text = "Form1";
        this.groupBox1.ResumeLayout(false);
        this.groupBox1.PerformLayout();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

#endregion

private System.Windows.Forms.Button btnStartSO;
private System.Windows.Forms.Button btnGenerateStatistics;
private System.Windows.Forms.Button btnRetrieveStatistics;
private System.Windows.Forms.TextBox txtStatisticName;
private System.Windows.Forms.TextBox txtRetrievedStatistics;
private System.Windows.Forms.Button btnRetrieveStatistic;
private System.Windows.Forms.GroupBox groupBox1;
}

}

```

C#

```

// File STATISTICSSO.CS
using System;
using System.Collections.Generic;
using System.Text;

using Microsoft.PointOfService;
using Microsoft.PointOfService.BaseServiceObjects;

[assembly: PosAssembly("Service Object Contractors, Inc.")]

namespace SOSamples.Statistics
{
    [ServiceObject(DeviceType.Msr,
        "SampleStatistics",
        "Sample Statistics Service Object",
        1,
        9)]

    public class StatisticsTest : MsrBase
    {
        // This will be incremented every time DeviceEnabled
        // is set so that different sets of demonstration
        // statistics can be generated.
        int enableCount = 0;

        // The name of a custom created statistic used to demonstrate
        // custom Statistic handlers.
        private const string PollingStatistic = "Polling Interval";

        // Statistic used to demonstrate IncrementStatistic.
        private const string TestIncrement = "MyIncrementableStat";
    }
}

```

```
// String returned from CheckHealth
private string MyHealthText;

public StatisticsTest()
{
    DevicePath = "Sample Statistics";
}

// This is the delegate which will be called for each
// statistic that we have associated with this delegate by
// calling SetStatisticHandler().
//
// Delegates like this should most commonly be used
// to get and set statistics in hardware. The delegate
// allows the POS for .NET statistic subsystem to query
// the value of a statistic in real time, before it is
// returned to the application.
private string GetHardwareInfo(string name)
{
    // Add your code to query values from hardware here.

    // Very simple demonstration: just return the name
    // of the statistic as its value;
    return name;
}

// In a typical Service Object implementation, statistics
// will be modified throughout the Service Object code. This
// method demonstrates the methods used to modify statistic
// values.
public void SetDemonstrationStatistics()
{
    // IncrementStatistic can be used to easily
    // increment a numeric statistic.
    IncrementStatistic(TestIncrement);

    switch(enableCount)
    {
        case 0:
            SetStatisticValue(StatisticManufacturerName,
                "Microsoft Corporation");
            break;
        case 1:
            SetStatisticValue(StatisticManufacturerName,
                "Service Control Contractors, Inc.");
            break;
        case 2:
            SetStatisticValue(StatisticManufacturerName,
                "Point of Service Controls .com");
            break;
    }

    if (++enableCount == 3 )
    {
```

```
        enableCount = 0;
    }
}

#region PosCommon overrides
// Returns the result of the last call to CheckHealth()

public override void Open()
{
    base.Open();

    // In your implementation of Open(), your Service Object
    // code should:
    // 1. Initialize statistics.
    // 2. Create custom statistics.
    // 3. Set statistic handlers for hardware Statistics.

    // 1. Initialize statistics
    SetStatisticValue(StatisticManufacturerName,
                      "Microsoft Corporation");
    SetStatisticValue(StatisticManufactureDate,
                      "2004-10-23");
    SetStatisticValue(StatisticModelName,
                      "Statistic Sample");
    SetStatisticValue(StatisticMechanicalRevision,
                      "1.0");
    SetStatisticValue(StatisticInterface,
                      "USB");

    // 2a. Create a new statistic to test Increment
    // method. No custom handler needed.
    CreateStatistic(TestIncrement, false, "blobs");

    // 2b. Create a new manufacturer statistic to demonstrate
    // custom attributes with StatisticHandlers.
    CreateStatistic(PollingStatistic, false,
                    "milliseconds");

    // 3. Set handlers for statistics stored in hardware.
    SetStatisticHandlers(PollingStatistic,
                         new GetStatistic(GetHardwareInfo), null);
    SetStatisticHandlers(StatisticSerialNumber,
                         new GetStatistic(GetHardwareInfo), null);
}

public override bool DeviceEnabled
{
    get
    {
        return base.DeviceEnabled;
    }
    set
    {
        // This method will set various statistics to
        // demonstrate the statistic APIs. We are going
```

```

        // to change statistic values every time the device
        // is enabled. This operation is just for
        // demonstration and would not be found in live code.
        SetDemonstrationStatistics();

        base.DeviceEnabled = value;
    }
}

public override string CheckHealthText
{
    get
    {
        // MsrBasic.VerifyState(musBeClaimed,
        // mustBeEnabled). This may throw an exception
        VerifyState(false, false);

        return MyHealthText;
    }
}

public override string CheckHealth(
    HealthCheckLevel level)
{
    // Verify that device is open, claimed, and enabled.
    VerifyState(true, true);

    // Your code here:
    // check the health of the device and return a
    // descriptive string.

    // Cache result in the CheckHealthText property.
    MyHealthText = "Ok";
    return MyHealthText;
}

public override DirectIOData DirectIO(
    int command,
    int data,
    object obj)
{
    // Verify that the device is open.
    VerifyState(false, false);

    return new DirectIOData(data, obj);
}
#endifregion PosCommon overrides

#region MsrBasic Overrides
protected override MsrFieldData ParseMsrFieldData(
    byte[] track1Data,
    byte[] track2Data,
    byte[] track3Data,
    byte[] track4Data,
    CardType cardType)

```

```
    {
        // Your code here:
        // Implement this method to parse track data
        // into fields which will be returned as
        // properties to the application (e.g., FirstName,
        // AccountNumber, etc.)
        return new MsrFieldData();
    }

    protected override MsrTrackData ParseMsrTrackData(
        byte[] track1Data,
        byte[] track2Data,
        byte[] track3Data,
        byte[] track4Data,
        CardType cardType)
{
    // Your code here:
    // Implement this method to convert raw track data.
    return new MsrTrackData();
}
#endregion MsrBasic Overrides
}
}
```

Compiling the Code

- For additional information about creating and compiling a Service Object project, see [Service Object Samples: Getting Started](#).

See Also

Other Resources

- [Developing a Custom Service Object](#)
- [System Configuration](#)

Base Class DirectIO Method (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The **DirectIO** method and the **DirectIOEvent** event are used to provide functionality to the application that is not otherwise supported by the standard Unified Point Of Service (UnifiedPOS) specification for a particular device type.

DirectIO Method

If a device has features that are not supported by the standard UnifiedPOS specification, a Service Object may implement a **DirectIO** method to give the application access to those features.

An example might be a **LineDisplay** device that supports multicolor output. Few, if any, **LineDisplay**-type devices support color output, but an independent hardware vendor (IHV) might produce such a device and want to have the new features available to applications.

Use of this method will make the application nonportable, since the implementation of the **DirectIO** method is vendor-specific. An application that uses a **DirectIO** method on Vendor A's **LineDisplay** device cannot depend on using a Vendor B's device.

DirectIOEvent

This event can be used to send vendor-specific information directly to the application. This event provides a means for a vendor-specific UnifiedPOS service to provide events to the application that are not otherwise supported by the UnifiedPOS control.

Using this event will make an application incompatible with devices from other vendors.

See Also

Other Resources

- [Developing Service Objects Using Base Classes](#)

Capability Properties (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Certain properties cannot be set directly within a Service Object. This comes up most often in the case of capability properties; those with the **Cap** prefix in their names. According to the Unified Point Of Service (UnifiedPOS) specification, these properties must be read-only; therefore, an implementation-specific mechanism is needed for the Service Object to change the value of these properties.

BaseClass Properties

Microsoft Point of Service for .NET (POS for .NET) **Base** classes have a protected property, **Properties**, for this purpose. This property returns a helper class which has writable versions of the read-only properties implemented in the **Base** class. For example, [PinPadBase](#) has a property called **Properties** that returns an object of type [PinPadProperties](#). And this object contains properties used to set various **PinPad**-specific capability properties, such as [CapDisplay](#).

PosCommon Properties

In addition to device-specific property classes, all POS for .NET **Base** and **Basic** classes also have a protected property called **CommonProperties** which returns an object of type [CommonProperties](#). This helper class is used to modify capability and status properties found in **PosCommon**.

Setting Properties Using Helper Classes

In general, a Service Object should always access the value of its common and class-specific properties using the helper classes. These properties may be written to by the Service Object and always contain the appropriate values.

The Service Object developer should be aware of what the POS for .NET framework may do when a particular value is changed. For example, the Service Object should generally not change **CommonProperties.State** since this may interfere with the POS for .NET internal state. Similarly, the Service Object developer should be aware that changing **CommonProperties.PowerState** may send a **StatusUpdateEvent** event to the application.

Note

When deriving from the POS for .NET **Base** or **Basic** classes, the Service Object should generally not change the value of **CommonProperties.State** to **ControlState.Closed**. Doing so prevents cleanup of the event queue, and POS for .NET may later throw exceptions as it tries to process events already in the queue.

See also

- [PinPadProperties](#)
- [StatusUpdateEventHandler](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

Manually manage your POS for .NET devices (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

You can manually perform most Microsoft Point of Service for .NET (POS for .NET) device management tasks without using the POS device manager (posdm.exe).

Manually managing POS for .NET devices

You can manually edit the POS for .NET configuration XML file to replicate most of the functionality available with posdm.exe.

You can find the location of the POS for .NET configuration XML file in the **Configuration** value under the registry key **HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\POSfor.NET**.

The default location for the configuration file is *%ProgramData%*Microsoft\Point Of Service\Configuration\Configuration.xml

The following table lists posdm.exe commands and the equivalent XML that you must add to the configuration xml file.

Posdm.exe command	Description	Configuration.xml	Example
ADDDEVICE	Add a physical non-Plug and Play device.	<pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="yes" PnP="no"> </Device> </ServiceObject> </PointOfServiceConfig></pre>	<p>Posdm.exe command:</p> <pre>Posdm ADDDEVICE COM1 /SONAME:"Microsoft Msr Simulator" /Type:msr</pre> <p>Configuration.xml:</p> <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr"> <Device> </Device> </ServiceObject> </PointOfServiceConfig></pre>

```
HardwarePath="COM1"  
Enabled="yes"  
PnP="no">  
  
</Device>  
  
</ServiceObject>  
</PointOfServiceConfig>
```

ADDNAME	Add a name to a device's list of names.	<pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="yes" PnP="no"> <LogicalName Name="[Device Name]" /> </Device> </ServiceObject> </PointOfServiceConfig></pre>	Posdm.exe command: <code>Posdm ADDNAME MyName /SONAME:"Microsoft Msr Simulator" /Path:COM1</code> Configuration.xml: <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr"> <Device HardwarePath="COM1" Enabled="yes" PnP="no"> <LogicalName Name="MyName" /> </Device> </ServiceObject> </PointOfServiceConfig></pre>
---------	---	--	--

ADDPROPERTY	Add a property to a device.	<pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="yes" PnP="no"> <Property Name="[Property Name]" Value="[Property Value]" /> </Device> </ServiceObject> </PointOfServiceConfig></pre>	Posdm.exe command: <pre>Posdm addproperty MyProperty MyValue /Name:MyName</pre> Configuration.xml: <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr" > <Device HardwarePath="COM1" Enabled="yes" PnP="no"> <LogicalName Name="MyName" /> <Property Name="MyProperty" Value="MyValue" /> </Device> </ServiceObject> </PointOfServiceConfig></pre>
DELETEDEVICE	Delete a physical non-Plug and Play device.	Remove the <Device> node.	
DELETENAME	Delete a name from	Remove the <LogicalName> node.	

	a device's list of names		
DELETEPROPERTY	Delete a property from a device.	Remove the <Property> node.	
DISABLE	Disable an SO on a POS device.	<p>Set <code>Enabled="no"</code> and <code>Default="no"</code> on the <Device> node.</p> <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="no" PnP="no" Default="no"> </Device> </ServiceObject> </PointOfServiceConfig></pre>	<p>Posdm.exe command:</p> <pre>Posdm disable /Path:COM1</pre> <p>Configuration.xml:</p> <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr"> <Device HardwarePath="COM1" Enabled="no" PnP="no" Default="no"> </Device> </ServiceObject> </PointOfServiceConfig></pre>
ENABLE	Enable an SO on a POS device.	Set <code>Enabled="yes"</code> on the <Device> node.	
INFO	Displays the device properties.	N/A	
LISTDEVICES	List the POS	N/A	

		devices on the target <host>.	
LISTNAMES	List the names associated with POS devices.	N/A	
LISTPROPS	List the properties associated with a device.	N/A	
LISTSOS	List the POS service objects on the target <host>.	<p>The service object search paths are all of the values under the registry key:</p> <p>HKLM\Software\Wow6432Node\Posfor.NET\ControlAssemblies</p> <p>The default search path is:</p> <pre>%CommonProgramFiles(x86)%\Microsoft Shared\Point Of Service\Control Assemblies\</pre> <p>POS for .NET will attempt to load all service object DLL's found in these paths.</p>	
SETDEFAULT	Set one device as the default of its <type>.	<p>Set <code>Default="yes"</code> on the <Device> node.</p> <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="yes" PnP="no" Default="yes"> </Device> </ServiceObject> </PointOfServiceConfig></pre>	<p>Posdm.exe command:</p> <pre>Posdm SETDEFAULT ON /Path:COM1</pre> <p>Configuration.xml:</p> <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr"> <Device HardwarePath="COM1" Enabled="yes" PnP="no" Default="yes"> </Device> </ServiceObject> </PointOfServiceConfig></pre>

```
es">  
</Device>  
</ServiceObject>  
</PointOfServiceConfig>
```

SETPATH	Sets the non-Plug and Play POS device <path>.	<pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="[Service Object Name]" Type="[Device Type]"> <Device HardwarePath="[Hardware Path]" Enabled="yes" PnP="no"> </Device> </ServiceObject> </PointOfServiceConfig></pre>	Posdm.exe command: <pre>Posdm SETPATH COM2 /SONAME:"Microsoft Msr Simulator" /Type:msr</pre> Configuration.xml: <pre><PointOfServiceConfig Version="1.0"> <ServiceObject Name="Microsoft Msr Simulator" Type="Msr"> <Device HardwarePath="COM2" Enabled="yes" PnP="no"> </Device> </ServiceObject> </PointOfServiceConfig></pre>
---------	---	---	--

POS Device Manager (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

With Microsoft Point of Service for .NET (POS for .NET), you can manage your POS devices in various ways, including:

- Adding and deleting POS devices from computers (non-Plug and Play devices only).
- Listing POS devices and/or Service Objects for particular computers.
- Configuring a POS for .NET Service Object (SO) to run for a particular device port (non-Plug and Play devices only).
- Setting a default device for a POS device class.
- Preventing a Service Object from running for a device.
- Assigning a logical name by which a POS application can access the Service Object for a device.

POS devices are managed via a Windows Management Instrumentation (WMI) provider installed on the POS device host as part of the POS for .NET installation. Instructions in this Guide assume that you have already installed POS for .NET.

You can access this WMI provider through the WMI API or by using the command-line tool included in POS for .NET, POSDM.EXE.

In This Section

- [Configure a device for remote management](#) Describes how you can configure devices to enable remote management.
- [Using the WMI API to Manage Devices](#) Describes how to use WMI tools to manage your POS devices.
- [Using Visual Studio .NET Management Extensions and the POS for .NET WMI Management Classes](#) Describes how to use Microsoft Visual Studio 2013 and the POS for .NET WMI management classes with your POS devices.
- [Using the POS Device Manager Command-Line Tool](#) Demonstrates how to use the POSDM command line to enable remote device management.

Related Sections

- [POS for .NET v1.14 Features](#) Provides a high-level overview of the Microsoft Point of Service for .NET (POS for .NET) architecture.
- [Developing a POS Application](#) Outlines POS for .NET key concepts and features that are useful to programmers developing POS applications.
- [Developing a Custom Service Object](#) Demonstrates how to create POS for .NET applications which use Service Objects to communicate with hardware devices.

Configure a device for remote management (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Learn how to configure your Microsoft Point of Service for .NET (POS for .NET) v1.14 device to enable remote management.

If you have trouble remotely connecting to your device, you may have to configure your device for remote management. If your device is joined to a domain, some or all of the following configurations may already be configured through Group Policy settings.

Configure a device for remote management

To enable remote management by using a local administrator account

1. Sign in to the device with an administrator account.
2. Set the following registry key to disable User Account Control remote restrictions:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\system]  
"LocalAccountTokenFilterPolicy"=dword:00000001
```

For more information about how to change this registry key, see [Description of User Account Control and remote restrictions](#).

3. Restart your device.

To enable Windows Management Instrumentation (WMI) traffic through a firewall

1. On the Start menu, right-click Command Prompt and then click Run as administrator.

2. To establish a Windows Firewall exception for WMI traffic, type the following command:

```
netsh advfirewall firewall set rule group="windows management instrumentation  
(wmi)" new enable=yes
```

 **Important**

When running ELM on an OS that uses a language other than English, use the localized group name.

3. (Optional) If ELM displays an error message that WMI did not respond or failed to connect, you can use individual commands for DCOM, WMI service, callback sink, and outgoing connections to enable WMI traffic.

- To establish a Windows Firewall exception for DCOM port 135, type the following command:

```
netsh advfirewall firewall add rule dir=in name="DCOM"  
program=%systemroot%\system32\svchost.exe service=rpcss action=allow  
protocol=TCP localport=135
```

- To establish a Windows Firewall exception for the WMI service, type the following command:

```
netsh advfirewall firewall add rule dir=in name ="WMI"  
program=%systemroot%\system32\svchost.exe service=winmgmt action = allow  
protocol=TCP localport=any
```

- To establish a Windows Firewall exception for the sink that receives callbacks from a remote computer, type the following command:

```
netsh advfirewall firewall add rule dir=in name ="UnsecApp"  
program=%systemroot%\system32\wbem\unsecapp.exe action=allow
```

- To establish a Windows Firewall exception for outgoing connections to a remote computer that the local computer is communicating with asynchronously, type the following command:

```
netsh advfirewall firewall add rule dir=out name ="WMI_OUT"  
program=%systemroot%\system32\svchost.exe service=winmgmt action=allow  
protocol=TCP localport=any
```

For more information about how to enable WMI traffic, see [Connecting to WMI Remotely](#) on MSDN.

See Also

Other Resources

- [POS Device Manager](#)

Using the WMI API to Manage Devices (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The WMI provider serves a WMI namespace called /root/MicrosoftPointOfService. This namespace defines four classes:

- [ServiceObject Class](#)
- [PosDevice Class](#)
- [LogicalDevice Class](#)
- [DeviceProperty Class](#)

ServiceObject represents a POS for .NET Service Object from a management perspective. **PosDevice** represents the physical device serviced by the Service Object. **LogicalDevice** represents a logical name assigned to a **PosDevice**, providing third-party applications with the ability to access a Service Object without conflicting with other applications that may also be accessing the same Service Object. **DeviceProperty** instances are name/value pairs that can be associated with a **PosDevice** to store optional configuration data for Service Objects.

In This Section

- [ServiceObject Class](#) Lists the properties and methods of the **ServiceObject** Class.
- [PosDevice Class](#) Lists the properties and methods of the **PosDevice** Class.
- [LogicalDevice Class](#) Lists the properties and methods of the **LogicalDevice** Class.
- [DeviceProperty Class](#) Lists the properties and methods of the **DeviceProperty** Class.
- [Using VBScript to Manage Devices](#) Demonstrates how to programmatically manage devices using the WMI API.

See Also

Other Resources

- [POS Device Manager](#)

ServiceObject Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The **ServiceObject** class represents the management view of a Service Object.

Properties

Name	Description
Name	Represents the name of the Service Object in string format.
Type	Represents the type of the POS device class that is implemented by the Service Object.
UposVersion	Represents version of the UPOS standard that the Service Object is implementing in string format.
Path	Represents the path of the Service Object's assembly as a string.
Version	The version number of the Service Object's assembly in string format.
Compatibility	The major version of POS for .NET with which the Service Object is compatible.
Description	A short string describing the Service Object.
IsPlugNPlay	A Boolean indicator of whether the Service Object supports Plug and Play.
IsLegacy	A Boolean indicator of whether the device is using a legacy (OPOS) Service Object.

Methods

Name	Description
AddDevice	<p>Adds a non-Plug and Play device for this Service Object.</p> <p>Accepts one string parameter, Path, which is the hardware path of the non-Plug and Play device to add. There is no return value.</p>
DeleteDevice	<p>Deletes a non-Plug and Play device associated with this Service Object.</p> <p>Accepts one string parameter, Path, which is the hardware path of the non-Plug and Play device to delete. There is no return value.</p>

See Also

Tasks

- [Using VBScript to Manage Devices](#)

Other Resources

- [Using the WMI API to Manage Devices](#)

PosDevice Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The **PosDevice** class represents a single physical POS device. The class provides properties and methods that are needed to manage that physical device.

Properties

Name	Description
Type	String representation of the POS device type or category.
SoName	The name of the Service Object for this physical device, in string format.
Path	The hardware path of a device, in string format. For Plug and Play devices, this path comes from the Plug and Play engine. For non-Plug and Play devices, it is provided via the AddDevice method of ServiceObject. For devices using legacy (OPOS) Service Objects, this may be blank.
HardwareDescription	The device description of the logical device, returned from the registry in string format and used by the Plug and Play engine. This may be blank for devices using legacy (OPOS) Service Objects.
IsPlugNPlay	A Boolean indicator of whether the device supports Plug and Play.
IsLegacy	A Boolean indicator of whether the device is using a legacy (OPOS) Service Object.
Enabled	A Boolean representation of whether the device is enabled or not. This property allows write access.
Default	A Boolean representation of whether the device is the default device in a POS device category. This property allows write access.

Methods

Name	Description

Name	Description
AddName	<p>Adds a logical name for the device.</p> <p>Accepts one string parameter, <i>Name</i>, which is the name of the logical device to add. The name must be unique within a device class (type). There is no returned value.</p> <p>Logical names are represented by the LogicalDevice class.</p>
Deletename	<p>Deletes the logical name from the device.</p> <p>Accepts one string parameter, <i>Name</i>, which is the name of the logical device to delete. There is no returned value.</p> <p>Logical names are represented by the LogicalDevice class.</p>
AddProperty	<p>Adds a property (a name/value pair) to this device.</p> <p>Accepts two string parameters, <i>Name</i>, which is the name of the property, and <i>Value</i>, which is the value of the property. There is no returned value.</p> <p>Device properties are represented by the Property class.</p>
DeleteProperty	<p>Deletes a property from this device.</p> <p>Accepts one string parameter, <i>Name</i>, which is the name of the property to be deleted. There is no return value.</p> <p>Device properties are represented by the Property class.</p>

See Also

Tasks

- [Using VBScript to Manage Devices](#)

Other Resources

- [Using the WMI API to Manage Devices](#)

LogicalDevice Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The **LogicalDevice** class represents a logical device associated with a **PosDevice**. It provides a naming mechanism so that applications can be developed independently and refers to the same device without conflict.

Properties

Name	Description
Type	String representing the POS device category that the logical device belongs to.
SoName	String representing the name of the Service Object.
Path	String representing the path of the physical device.
Name	String representing the name for the logical device.

See Also

Tasks

- [Using VBScript to Manage Devices](#)

Other Resources

- [Using the WMI API to Manage Devices](#)

DeviceProperty Class (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The **DeviceProperty** class represents a name/value pair of a configuration property for a physical device. There may be multiple **DeviceProperty** classes associated with a **PosDevice**.

Properties

Name	Description
Type	String representing the POS device category.
SoName	String representing the name of the Service Object.
Path	String representing the path of the physical device.
Name	String representing the name of this property.
Value	String representing the data of this property.

See Also

Tasks

- [Using VBScript to Manage Devices](#)

Concepts

- [PosDevice Class](#)

Other Resources

- [Using the WMI API to Manage Devices](#)

Using VBScript to Manage Devices (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

Using the WMI API documented in this section, it is possible to manage devices using managed code or scripting. **POSMDM.EXE** is a command-line interface to this API. This VBScript sample does the following:

- It uses the WMI method **ExecQuery** to retrieve a list of installed **PosDevice** objects. With this list of Service Objects, the script displays their type, name, corresponding path, and their enabled or disabled status. This is analogous to running the following command:

```
PosDM.exe LISTDEVICES
```

- It then attempts to assign the path **COM1** to the installed Service Object, **Microsoft Msr Simulator** using the **AddDevice** method. This is equivalent to running:

```
PosDM.exe ADDDEVICE COM1 /SONAME:Microsoft Msr Simulator
```

- If the **AddDevice** method fails, the script catches the error and assumes that **COM1** may have already been added to the device and therefore attempts to delete it by calling **DeleteDevice**. This is equivalent to running:

```
PosDM.exe DELETEDevice COM1
```

- If the **AddDevice** method had previously failed, the script then attempts to call **AddDevice** again. The program exits if the method fails.
- Finally, the sample attempts to add the logical name **MSRSim** to this Service Object by calling **AddName**. This is equivalent to running:

```
PosDM.exe ADDNAME MSRSim /SONAME:"Microsoft Msr Simulator"
```

It is possible to see the results of this sample by running:

```
PosDM.exe LISTDEVICES
```

And

```
PosDM.exe LISTNAMES
```

To run the sample

1. The Service Object **Microsoft Msr Simulator** was installed with the SDK. Be sure that it is installed on the computer which you will be using to run the sample.
2. Copy this script to a file **PosDMSample.vbs**
3. Execute the script with the following command line:

```
CScript //U PosDMSample.vbs
```

Example

Visual Basic Script

```
'Get a handle to the POS namespace service into 'objServices'.
Set objLocator = CreateObject("WbemScripting.SWbemLocator")
Set objServices = objLocator.ConnectServer(
    "/root/MicrosoftPointOfService")

'List the POS devices.
EnumeratePosDevice

'Add a name: MSRSim for Msr Simulator by retrieving the SO and invoking
AddDevice() then AddName()
WScript.Echo "Add Device on COM1 and add name 'MSRSim' for MsrSimulator ..."
Set objSO = objServices.Get("ServiceObject.Type='Msr',Name='Microsoft Msr
Simulator'")

On Error Resume Next
objSO.AddDevice "COM1"
if Err.number <> 0 Then
    WScript.Echo "AddDevice failed - it already is in use."
    WScript.Echo "Try to delete the device..."

On Error Resume Next
objSO.DeleteDevice "COM1"
if Err.number <> 0 Then
    WScript.Echo "DeleteDevice failed"
    WScript.Quit 1
end if

WScript.Echo "DeleteDevice succeeded! Attempting AddDevice again..."

On Error Resume Next
objSO.AddDevice "COM1"
if Err.number <> 0 Then
    WScript.Echo "AddDevice failed a second time - exiting"
    WScript.Quit 2
end if
```

```

end if

Set objDevice = objServices.Get("PosDevice.SoName='Microsoft Msr
Simulator',Type='Msr',Path='COM1'")
objDevice.AddName "MSRSim"
Set objDevice = GetDevice("Msr", "MSRSim")
WScript.Echo "Added 'MSRSim' to: " & objDevice.Type & vbTab &
objDevice.SoName & vbTab & objDevice.Path

'Enumerate the sClass by name
Sub EnumeratePosDevice( )
    sClass = "PosDevice"
    WScript.Echo "Enumerating " & sClass & "..." & vbCrLf

    Set collection = objServices.ExecQuery("SELECT * From " & sClass)
    For Each obj In collection
        Enabled = "DISABLED"
        if obj.Enabled = true Then
            Enabled = "ENABLED"
        end If
        WScript.Echo obj.Type & Space(15-len(obj.type)) & obj.SoName &
Space(35-len(obj.SoName)) & Enabled & vbTab & obj.Path
    Next
    WScript.Echo vbCrLf
End Sub

'Return a PosDevice matching DeviceType and Name.
Function GetDevice( DeviceType, Name )
    Set Logical = GetLogicalDevice( DeviceType, Name )
    objectPath = "PosDevice.SoName=''' & Logical.SoName & ''',Type=''' &
DeviceType & ''',Path=''' & Logical.Path & '''"
    Set GetDevice = objServices.Get(objectPath)
End Function

'Return a LogicalDevice matching DeviceType and Name.
Function GetLogicalDevice( DeviceType, Name )
    Query = "SELECT * From LogicalDevice WHERE Type = '' & DeviceType & '' AND
Name=''' & Name & '''"
    Set collection = objServices.ExecQuery( Query )
    For Each obj In collection
        Set GetLogicalDevice = obj
        exit For
    Next
End Function

```

If the path COM1 has not been assigned to a device, the sample produces output similar to this code.

Console

```

Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

```

Enumerating PosDevice...

Msr	Microsoft Msr Simulator	ENABLED	
Msr	Microsoft Msr Simulator	ENABLED	COM1
Keylock	Microsoft Keylock Simulator	ENABLED	
Scanner	Microsoft Scanner Simulator	ENABLED	
CashDrawer	Microsoft CashDrawer Simulator	ENABLED	
CheckScanner	Microsoft CheckScanner Simulator	ENABLED	
LineDisplay	Microsoft LineDisplay Simulator	ENABLED	
PinPad	Microsoft PinPad Simulator	ENABLED	
PosPrinter	Microsoft PosPrinter Simulator	ENABLED	
PosKeyboard	Microsoft PosKeyboard Simulator	ENABLED	

Add Device on COM1 and add name 'MSRSim' for MsrSimulator ...

AddDevice failed - it already be in use.

Try to delete the device...

DeleteDevice succeeded! Attempting AddDevice again...

Added 'MSRSim' to: Msr Microsoft Msr Simulator

If the path COM1 is already in use and no other error occurs, the script produces output that looks like this code.

Console

Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Enumerating PosDevice...

Msr	Microsoft Msr Simulator	ENABLED	
Msr	Microsoft Msr Simulator	ENABLED	COM1
Keylock	Microsoft Keylock Simulator	ENABLED	
Scanner	Microsoft Scanner Simulator	ENABLED	
CashDrawer	Microsoft CashDrawer Simulator	ENABLED	
CheckScanner	Microsoft CheckScanner Simulator	ENABLED	
LineDisplay	Microsoft LineDisplay Simulator	ENABLED	
PinPad	Microsoft PinPad Simulator	ENABLED	
PosPrinter	Microsoft PosPrinter Simulator	ENABLED	
PosKeyboard	Microsoft PosKeyboard Simulator	ENABLED	

Add Device on COM1 and add name 'MSRSim' for MsrSimulator ...

AddDevice failed - it already be in use.

Try to delete the device...

DeleteDevice succeeded! Attempting AddDevice again...

Added 'MSRSim' to: Msr Microsoft Msr Simulator

See Also

Other Resources

- Using the WMI API to Manage Devices
- POS Device Manager

Using Visual Studio .NET Management Extensions and the POS for .NET WMI Management Classes (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

You can use Server Explorer in Microsoft Visual Studio 2013 to navigate the **Microsoft.PointOfService** namespace and drag the instances of the classes into your project's Class Designer.

This feature requires that Visual Studio 2013 and POS for .NET are installed on the local development computer.

To use the extension

1. Launch Visual Studio 2013 and, from the **VIEW** menu, open the **Server Explorer** window.
2. Expand the **Servers** node, and then expand the **Machine** node.
3. Right-click the **Management Classes** node and then click **Add Classes** on the shortcut menu.
4. In the **Add Classes** dialog box, expand the **root\MicrosoftPointOfService** node in the **Available Classes** tree view.
5. Select the **DeviceProperty** class, and then click **Add** to add the class to the Server Explorer. Repeat this step for the **LogicalDevice**, **PosDevice**, and **ServiceObject** classes.

To use the management classes

1. Create a .NET project.
2. Open the Server Explorer.
3. Right-click the **DeviceProperty** node, and then click **Generate Managed Class** on the shortcut menu to add the generated class to the project. Repeat this step for

the **LogicalDevice**, **PosDevice**, and **ServiceObject** classes to generate managed classes.

To use an instance of a management class

1. In the Server Explorer, expand the desired class to list the available class objects.
2. Drag the instances onto the projects class designer.

Example

The following code example demonstrates the use of the **PosDevice** class **GetInstances** method to enumerate Point of Service devices. It creates a collection of the devices within a scope. It then lists the type, name and path for each device in the collection and indicates whether the device is enabled or disabled.

C#

```
using System;
using System.Management;
using ROOT.MICROSOFTPOINTOFSERVICE;

namespace Management
{
    public class Test
    {
        public Test()
        {
            ManagementScope scope = new
ManagementScope("root\\microsoftpointofservice");
            scope.Connect();
            PosDevice.PosDeviceCollection devices =
PosDevice.GetInstances(scope, "");
            string format = "{0,10}\t{1,25}\t{2}\t{3,50}";
            if( devices.Count > 0 )
                Console.WriteLine(format, "Type", "Name", "Enabled", "Path");
            foreach( PosDevice d in devices )
            {
                Console.WriteLine(format, d.Type, d.SoName, d.Enabled ? 'Y' :
'N', d.Path);
            }
        }

        static int Main()
        {
            Test t = new Test();
            return 0;
        }
    }
}
```

```
    }  
}
```

See Also

Other Resources

- [POS Device Manager](#)
- [Using the POS Device Manager Command-Line Tool](#)
- [Using the WMI API to Manage Devices](#)

Using the POS Device Manager Command-Line Tool (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

You can use the POS Device Manager Command-Line Tool after Microsoft Point of Service for .NET (POS for .NET) is installed on your computer and after your devices are configured for remote management as described in [Configure a device for remote management](#).

On the command line, type **posdm** followed by a space and then any of the commands and switches listed below.

The general syntax is:

posdm [general switches] command [command arguments]

In This Section

- [Command-Line Help for POSDM](#) Demonstrates how to access Help about POSDM syntax, usage, and commands.
- [General POSDM Switches](#) Describes several commonly used POSDM command-line switches.
- [POSDM Commands](#) Provides a comprehensive list of the commands that you can use in POSDM.
- [POSDM Command Arguments](#) Describes several switches that can be used as arguments in POSDM.

See Also

Other Resources

- [POS Device Manager](#)

Command-Line Help for POSDM (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

You can display information about the general usage and syntax for POSDM.EXE by typing the following on the command line:

```
posdm /?
```

To get more information about any POSDM command, type the following on the command line:

```
posdm help command
```

Example:

```
posdm help listdevices
```

See Also

Concepts

- [POSDM Commands](#)

Other Resources

- [Using the POS Device Manager Command-Line Tool](#)

General POSDM Switches (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The following general switches may be used between POSDM and a command on the command line.

Switch	Description
/?	Displays posdm Help information.
/output:file	Prints all output to file instead of to the console.
/machine:host	Uses host as the remote computer. Default is the local computer.
/user:username	Uses user name as the user account. Prefix with domain separated by a backslash if the user domain is not the current one.
/password:password	Uses password with the specified user name account.

See Also

Concepts

- [POSDM Commands](#)

Other Resources

- [Using the POS Device Manager Command-Line Tool](#)

POSDM Commands (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

POSDM uses the following commands.

Command	Description	Syntax and Examples
adddevice	Adds a physical non-Plug and Play device.	<p><code>posdm [general switches] adddevicepath filter[/info]</code></p> <p>where <i>path</i> is a hardware path of the physical device,</p> <p><i>filter</i> is one or more of the following:</p> <p><code>/type:devicetype</code></p> <p><code>/soname:soname</code></p> <p>and</p> <p><code>/info</code> is a switch that displays all device properties.</p> <p>Example:</p> <p><code>posdm adddevice COM3 /soname:MsrSimulator</code></p> <p>This adds a device with the hardware path COM3 to the MsrSimulator Service Object.</p>
addname	Adds a logical name to a device.	<p><code>posdm [general switches] addname devicename filter</code></p> <p>where <i>devicename</i> is the logical name to give to the device, and <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <p><code>/type:devicetype</code></p> <p><code>/soname:soname</code></p> <p><code>/path:hardware_path</code></p> <p><code>/name:devicename</code></p> <p>Example:</p>

		<pre>posdm addname MainMSR /type:MSR /path:COM3</pre> <p>This adds the logical name MainMSR for the MSR device on the COM3 hardware path.</p> <pre>posdm addname BackupMSR /name:MainMSR</pre> <p>This adds the logical name BackupMSR for the device named MainMSR.</p>
addproperty	Adds a configuration property to a device.	<pre>posdm [general switches] addproperty<i>propertynam</i>e <i>value</i> <i>filter[/info]</i></pre> <p>where <i>propertynam</i>e is the name of the property and <i>value</i> is the initial value for that property, and <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <pre>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i> /name:<i>devicename</i></pre> <p><i>/info</i> is a switch that displays all device properties.</p> <p>Example:</p> <pre>posdm addproperty PrintSpecialGreeting "Happy New Year!" /name:MainMSR</pre> <p>This adds the PrintSpecialGreeting property with the value of "Happy New Year!" to the device named <i>MainMSR</i>.</p>
deletedevice	Deletes a physical non-Plug and Play device.	<pre>posdm [general switches] deletedevice[i<i>path</i>] <i>filter</i></pre> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <pre>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i></pre>

		<p>Example:</p> <p>posdm deletedevice COM3 /type:Msr</p> <p>This deletes the MSR device on COM3.</p> <p>Only devices previously added by the adddevice command can be deleted.</p>
deletename	Deletes a logical name from a device's list of names.	<p>posdm [general switches] deletename devicename<i>filter</i> [/info]</p> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <p><i>/type:devicetype</i></p> <p><i>/soname:soname</i></p> <p><i>/path:hardware_path</i></p> <p><i>/name:devicename</i></p> <p>and</p> <p><i>/info</i> is a switch that displays all device properties.</p> <p>Example:</p> <p>posdm deletename "Main Scanner" /type:Scanner /path:COM3</p> <p>This deletes the logical name Main Scanner for the scanner device on the COM3 path.</p> <p>Only logical names previously added by the addname command can be deleted.</p>
deleteproperty	Deletes a configuration property from a device.	<p>posdm [general switches] deleteproperty<i>propertyname filter</i> [/info]</p> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <p><i>/type:devicetype</i></p> <p><i>/soname:soname</i></p> <p><i>/path:hardware_path</i></p> <p><i>/name:devicename</i></p> <p>and</p>

		/info is a switch that displays all device properties.
		<pre>posdm deleteproperty PrintSpecialGreeting /name:MainMSR</pre> <p>This deletes the PrintSpecialGreeting property from the device named MainMSR.</p> <p>Only configuration properties previously added by the addproperty command can be deleted.</p>
disable	Prevents a Service Object from running for a physical POS device.	<pre>posdm [general switches] disable<i>filter</i></pre> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <pre>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i> /name:<i>devicename</i></pre> <p>Example:</p> <pre>posdm disable /name:ReceiptPrn</pre> <p>This prevents a Service Object from running for a device named ReceiptPrn. As a result, applications will not see the device in the list of available POS devices.</p>
enable	Permits a Service Object to run for a physical POS device.	<pre>posdm [general switches] enable<i>filter</i></pre> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <pre>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i> /name:<i>devicename</i></pre> <p>Example:</p> <pre>posdm enable /type:MSR</pre>

		This permits a Service Object to run for all MSR devices.
info	Displays information about the device, including its configuration properties.	<p>posdm [general switches] info<i>filter</i></p> <p>where filter is one or more of the following needed to uniquely identify a device:</p> <p><i>/type:devicetype</i></p> <p><i>/soname:soname</i></p> <p><i>/path:hardware_path</i></p> <p><i>/name:devicename</i></p> <p>Example:</p> <p>posdm info /name:MSR1</p> <p>This command displays information about a device with the logical name "MSR1."</p>
listdevices	Lists the physical POS devices.	<p>posdm [general switches] listdevices</p> <p>[/type:devicetype]</p> <p>where the <i>/type:devicetype</i> switch narrows the list to a particular type of device.</p> <p>Examples:</p> <p>posdm listdevices</p> <p>This displays a list of all physical POS devices installed on the local computer.</p> <p>posdm listdevices /type:MSR</p> <p>This displays a list of all MSR devices installed on the local computer.</p> <p>posdm /machine:Center10 /username:JohnDoe3 /password:B\$tg59ce listdevices</p> <p>This lists all physical POS devices installed on the computer named Center10, after logging on with username and password credentials.</p>
listnames	Lists the logical names associated with POS devices.	<p>posdm [general switches] listnames<i>filter</i></p> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p>

/type:devicetype

/soname:soname

/path:hardware_path

Example:

posdm listnames /type:MSR /path:COM3

This displays a list of names associated with the MSR device on COM3.

listprops	Lists the configuration properties associated with a POS device and their values.	posdm [general switches] listprops <i>filter</i> where filter is one or more of the following needed to uniquely identify a device:
-----------	---	---

/type:devicetype

/soname:soname

/path:hardware_path

/name:devicename

Example:

posdm listprops /type:MSR /path:COM3

This displays a list of property names and values associated with the MSR device on COM3.

listsos	Lists the POS Service Objects on the target machine.	posdm [general switches] listsos [/type:devicetype] where the /type:devicetype switch narrows the list to a particular type of device.
---------	--	---

Examples:

posdm /output:a:\solist.txt listsos

This writes a list of all Service Objects installed on the local computer to a file named solist.txt on drive A.

posdm listsos /type:MSR

This displays a list of all Service Objects associated with MSR devices on the local computer.

```
posdm /machine:Center10  
/username:JohnDoe3  
/password:B$tg59ce listsos
```

This lists all Service Objects on the computer named Center10, after logging on with username and password credentials.

setdefault	<p>Sets one device as the default of its <i>type</i>.</p> <p>The default flag directs the PosExplorer.GetDevice(<i>type</i>) method to return this device even if there is more than one device of the type available.</p>	<pre>posdm [general switches] setdefault ON OFF <i>filter</i> [/info]</pre> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <p>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i> /name:<i>devicename</i></p> <p>The /info switch causes all device properties to be displayed.</p> <p>Example:</p> <pre>posdm setdefault ON /name:FirstScanner</pre> <p>This example designates FirstScanner as the one the CCL device enumeration will find.</p>
setpath	<p>Sets the POS device path for non-Plug and Play devices.</p>	<pre>posdm [general switches] setpath <i>hardware_path filter</i></pre> <p>where <i>filter</i> is one or more of the following needed to uniquely identify a device:</p> <p>/type:<i>devicetype</i> /soname:<i>soname</i> /path:<i>hardware_path</i> /name:<i>devicename</i></p> <p>Example:</p> <pre>posdm setpath COM2 /type:MSR</pre> <p>This sets the hardware path for MSR devices to COM2.</p>

The **setpath** command works only for non-Plug and Play devices previously added with the **adddevice** command.

See Also

Concepts

- [POSDM Command Arguments](#)
- [Command-Line Help for POSDM](#)

Other Resources

- [Using the POS Device Manager Command-Line Tool](#)

POSDM Command Arguments (POS for .NET v1.14 SDK Documentation)

Article • 02/21/2023

The following switches can be used as arguments for POSDM commands.

Command Switch	Description
/type: <i>devicetype</i>	The device type as determined by the Service Object supplier. Example: /type:msr specifies a magnetic strip reader.
/soname: <i>soname</i>	The Service Object name from the Service Object supplier. Names containing spaces must be enclosed in double quotation marks (""). Example: /soname:"Super MSR" specifies a Service Object named Super MSR.
/path: <i>hardware_path</i>	The hardware path for the device. Example: /path:COM2 specifies a location of COM2.
/name: <i>devicename</i>	The logical name for the device. This is useful when there are multiple instances of the Service Object, and also provides a way for an application to refer to a Service Object without hard coding. This name must be unique within devicetype . If a name contains a space, it must be contained within double quotation marks (" "). Examples: /name:msr4 refers to a device with the logical name msr4, /name:"Main MSR" refers to a device with the logical name of Main MSR.

See Also

Concepts

- [POSDM Commands](#)

Other Resources

- [Using the POS Device Manager Command-Line Tool](#)

April 2024 security and quality rollup

Article • 07/03/2024

Released April 9, 2024

Summary of what's new in this release

- [Security improvements](#)
- [Quality and reliability improvements](#)

Security improvements

CVE-2024-21409 – Remote code execution vulnerability

This security update addresses a remote code execution vulnerability detailed in [CVE 2024-21409](#).

Defense in depth vulnerability

This security update addresses an issue where version of the OSS zlib library is out of date.

Defense in depth vulnerability

This security update addresses an issue in AIA fetching process.

Quality and reliability improvements

This release contains the following quality and reliability improvements.

ASP.NET

Addresses an issue with `JavaScriptSerializer` where after installing the January Security and Quality update, there is a performance degradation. (*Applies to: .NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1.*)

CLR

Addressed an issue with thread pool recycled lists becoming unresponsive on multi-CPU-group computers. (*Applies to: .NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1.*)

Addressed an issue where Interlocked.Read from 32-bit apps are much slower on some computers. (*Applies to: .NET Framework 4.8.1.*)

Compilers

Addresses an issue when the native C# compiler (csc.exe) is used to compile code making calls to COM interop assemblies. (*Applies to: .NET Framework 4.8.1.*)

Known issues

This release contains no known issues.

Summary tables

The following table outlines the updates in this release.

[] Expand table

Product version	Cumulative update
Microsoft server operating system, version 23H2	
.NET Framework 3.5, 4.8.1	5036617
Windows 11, version 22H2 and Windows 11, version 23H2	
.NET Framework 3.5, 4.8.1	5036620
Windows 11, version 21H2	5037037
.NET Framework 3.5, 4.8	5036611
.NET Framework 3.5, 4.8.1	5036619
Microsoft server operating system, version 22H2 and version 21H2	5037033
.NET Framework 3.5, 4.8	5036613
.NET Framework 3.5, 4.8.1	5036621
Windows 10, version 22H2	5037036
.NET Framework 3.5, 4.8	5036608

Product version	Cumulative update
.NET Framework 3.5, 4.8.1	5036618
Windows 10, version 21H2	5037035
.NET Framework 3.5, 4.8	5036608
.NET Framework 3.5, 4.8.1	5036618
Windows 10 1809 and Windows Server 2019	5037034
.NET Framework 3.5, 4.7.2	5036604
.NET Framework 3.5, 4.8	5036610
Windows 10 1607 and Windows Server 2016	
.NET Framework 3.5, 4.6.2, 4.7, 4.7.1, 4.7.2	5032197
.NET Framework 4.8	5036609
Windows 10 1507	
.NET Framework 3.5, 4.6, 4.6.2	5032199

The following table is for earlier Windows and Windows Server versions for Security and Quality Rollup updates.

[\[+\] Expand table](#)

Product version	Security and quality rollup
Windows Server 2012 R2	5037040
.NET Framework 3.5	5036627
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5036606
.NET Framework 4.8	5036614
Windows Server 2012	5037039
.NET Framework 3.5	5036624
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5036605
.NET Framework 4.8	5036612
Windows Server 2008 R2	5037038
.NET Framework 3.5.1	5036626

Product version	Security and quality rollup
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5036607 ↗
.NET Framework 4.8	5036615 ↗
Windows Server 2008	5037041 ↗
.NET Framework 2.0, 3.0	5036625 ↗
.NET Framework 3.5 SP1	5036637 ↗
.NET Framework 4.6.2	5036607 ↗

The following table is for earlier Windows and Windows Server versions for Security Only updates, which aren't cumulative.

[\[+\] Expand table](#)

Product version	Security only update
Windows Server 2008 R2	5037127 ↗
.NET Framework 3.5.1	5036634 ↗
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5036631 ↗
.NET Framework 4.8	5036632 ↗
Windows Server 2008	5037128 ↗
.NET Framework 2.0, 3.0	5036633 ↗
.NET Framework 3.5 SP1	5036636 ↗
.NET Framework 4.6.2	5036631 ↗

The operating system row lists a KB which will be used for update offering purposes. When the operating system KB is offered, the applicability logic will determine the specific .NET Framework update(s) will be installed. Updates for individual .NET Framework versions will be installed based on the version of .NET Framework that is already present on the device. Because of this the operating system KB is not expected to be listed as installed updates on the device. The expected update to be installed are the .NET Framework specific version updates listed in the preceding table.

 Collaborate with us on
GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

April 2024 cumulative update preview

Article • 07/03/2024

Released April 23, 2024

Summary of what's new in this release

- Security improvements
- Quality and reliability improvements

Security improvements

There are no new security improvements in this release. This update is cumulative and contains all previously released security improvements.

Quality and reliability improvements

This release contains the following quality and reliability improvements.

CLR

Addresses an issue where crashes can occur if several threads all concurrently query the `ITypeInfo` implementation of the same managed type. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Addresses an issue where `ISymUnmanagedReader::GetMethodsFromDocumentPosition` and `ISymUnmanagedReader2::GetMethodsInDocument` might return incorrect results under certain circumstances. (*Applies to: .NET Framework 4.8.1.*)

.NET libraries

Addresses an issue that can be triggered in the fbx file parser. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Addresses an issue to use MIST-validated implementations of FIPS algorithms. (*Applies to: .NET Framework 4.8, 4.8.1.*)

.NET fundamentals

Addresses an issue with wildcard format changes introduced in IIS 10. (*Applies to: .NET Framework 4.8, 4.8.1.*)

WPF

Addresses an issue where apps crash when calling the `GetWindowText` and `GetWindowTextLength` methods. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Known issues

This release contains no known issues.

Summary tables

The following table outlines the updates in this release.

 Expand table

Product version	Cumulative update preview
Windows 11, version 22H2 and Windows 11, version 23H2	
.NET Framework 3.5, 4.8.1	5037591 ↗
Windows 10, version 22H2	5037724 ↗
.NET Framework 3.5, 4.8	5037592 ↗
.NET Framework 3.5, 4.8.1	5037587 ↗

The operating system row lists a KB which will be used for update offering purposes. When the operating system KB is offered, the applicability logic will determine the specific .NET Framework update(s) will be installed. Updates for individual .NET Framework versions will be installed based on the version of .NET Framework that is already present on the device. Because of this the operating system KB is not expected to be listed as installed updates on the device. The expected update to be installed are the .NET Framework specific version updates listed in the table above.



Collaborate with us on
GitHub



.NET feedback

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET is an open source project.
Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

May 2024 security and quality rollup

Article • 07/03/2024

Released May 14, 2024

Summary of what's new in this release

- [Security improvements](#)
- [Quality and reliability improvements](#)

Security improvements

There are no new security improvements in this release. This update is cumulative and contains all previously released security improvements.

Quality and reliability improvements

This release contains the following quality and reliability improvements.

CLR

Addresses an issue where crashes can occur if several threads all concurrently query the `ITypeInfo` implementation of the same managed type. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Addresses an issue where `ISymUnmanagedReader::GetMethodsFromDocumentPosition` and `ISymUnmanagedReader2::GetMethodsInDocument` might return incorrect results under certain circumstances. (*Applies to: .NET Framework 4.8.1.*)

.NET libraries

Addresses an issue that can be triggered in the fbx file parser. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Addresses an issue to use MIST-validated implementations of FIPS algorithms. (*Applies to: .NET Framework 4.8, 4.8.1.*)

.NET fundamentals

Addresses an issue with wildcard format changes introduced in IIS 10. (*Applies to: .NET Framework 4.8, 4.8.1.*)

WPF

Addresses an issue where apps crash when calling the `GetWindowText` and `GetWindowTextLength` methods. (*Applies to: .NET Framework 4.8, 4.8.1.*)

Known issues

This release contains no known issues.

Summary tables

 Expand table

Product version	Cumulative update
Microsoft server operating system, version 23H2	
.NET Framework 3.5, 4.8.1	5038075 ↗
Windows 11, version 22H2 and Windows 11, version 23H2	
.NET Framework 3.5, 4.8.1	5037591 ↗
Windows 11, version 21H2	5038286 ↗
.NET Framework 3.5, 4.8	5037934 ↗
.NET Framework 3.5, 4.8.1	5037931 ↗
Microsoft server operating system, version 22H2 and version 21H2	5038282 ↗
.NET Framework 3.5, 4.8	5037930 ↗
.NET Framework 3.5, 4.8.1	5037929 ↗
Windows 10, version 22H2	5038285 ↗
.NET Framework 3.5, 4.8	5037592 ↗
.NET Framework 3.5, 4.8.1	5037587 ↗
Windows 10, version 21H2	5038284 ↗
.NET Framework 3.5, 4.8	5037592 ↗

Product version	Cumulative update
.NET Framework 3.5, 4.8.1	5037587
Windows 10 1809 and Windows Server 2019	5038283
.NET Framework 3.5, 4.7.2	5037932
.NET Framework 3.5, 4.8	5037933
Windows 10 1607 and Windows Server 2016	
.NET Framework 3.5, 4.6.2, 4.7, 4.7.1, 4.7.2	5037763
.NET Framework 4.8	5037926

The following table is for earlier Windows and Windows Server versions for Security and Quality Rollup updates.

[Expand table](#)

Product version	Security and quality rollup
Windows Server 2012 R2	5037040
.NET Framework 3.5	5036627
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5037925
.NET Framework 4.8	5037923
Windows Server 2012	5037039
.NET Framework 3.5	5036624
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5037924
.NET Framework 4.8	5037922
Windows Server 2008 R2	5037038
.NET Framework 3.5.1	5036626
.NET Framework 4.6.2, 4.7, 4.7.1, 4.7.2	5037917
.NET Framework 4.8	5037916
Windows Server 2008	5038291
.NET Framework 2.0, 3.0	5036625
.NET Framework 3.5 SP1	5036637

Product version	Security and quality rollup
.NET Framework 4.6.2	5037917

The operating system row lists a KB which will be used for update offering purposes. When the operating system KB is offered, the applicability logic will determine the specific .NET Framework update(s) will be installed. Updates for individual .NET Framework versions will be installed based on the version of .NET Framework that is already present on the device. Because of this the operating system KB is not expected to be listed as installed updates on the device. The expected update to be installed are the .NET Framework specific version updates listed in the table above.

 [Collaborate with us on GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

 .NET

[.NET feedback](#)

.NET is an open source project.
Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

June 2024 cumulative update preview

Article • 07/03/2024

Released June 25, 2024

Summary of what's new in this release

- Security improvements
- Quality and reliability improvements

Security improvements

There are no new security improvements in this release. This update is cumulative and contains all previously released security improvements.

Quality and reliability improvements

.NET fundamentals

Addresses an issue with x509 certificate use under PPL in Azure AD

Winforms

Addresses an issue with the size of memory leaks associated with AccessibleObjects held in memory due to ref-counting

Known Issues in this Release

This release contains no known issues.

Summary tables

The following table outlines the updates in this release.

[+] Expand table

Product version	Cumulative update preview
Windows 11, version 22H2 and Windows 11, version 23H2	
.NET Framework 3.5, 4.8.1	5039866 ↗
Windows 10, version 22H2	5040370 ↗
.NET Framework 3.5, 4.8	5039867 ↗
.NET Framework 3.5, 4.8.1	5039865 ↗

The operating system row lists a KB which will be used for update offering purposes. When the operating system KB is offered, the applicability logic will determine the specific .NET Framework update(s) will be installed. Updates for individual .NET Framework versions will be installed based on the version of .NET Framework that is already present on the device. Because of this the operating system KB is not expected to be listed as installed updates on the device. The expected update to be installed are the .NET Framework specific version updates listed in the table above.

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

What's new in .NET Framework

Article • 06/01/2023

ⓘ Note

.NET Framework [is serviced monthly](#) with security and reliability bug fixes. .NET Framework will continue to be included with Windows, with no plans to remove it. You don't need to migrate your .NET Framework apps, but for new development, use [.NET 6 or later](#).

This article summarizes key new features and improvements in the following versions of .NET Framework:

- [.NET Framework 4.8.1](#)
- [.NET Framework 4.8](#)
- [.NET Framework 4.7.2](#)
- [.NET Framework 4.7.1](#)
- [.NET Framework 4.7](#)
- [.NET Framework 4.6.2](#)
- [.NET Framework 4.6.1](#)
- [.NET 2015 and .NET Framework 4.6](#)
- [.NET Framework 4.5.2](#)
- [.NET Framework 4.5.1](#)
- [.NET Framework 4.5](#)

This article does not provide comprehensive information about each new feature and is subject to change. For general information about .NET Framework, see [Getting Started](#). For supported platforms, see [System Requirements](#). For download links and installation instructions, see [Installation Guide](#).

ⓘ Note

The .NET Framework team also releases features out of band, using NuGet, to expand platform support and introduce new functionality, such as immutable collections and SIMD-enabled vector types. For more information, see [Additional Class Libraries and APIs](#) and [.NET Framework and Out-of-Band Releases](#). See a [complete list of NuGet packages](#) for .NET Framework.

Introducing .NET Framework 4.8.1

.NET Framework 4.8.1 builds on previous versions of .NET Framework 4.x by adding many new fixes and several new features while remaining a very stable product.

Download and install .NET Framework 4.8.1

You can download .NET Framework 4.8.1 from the following locations:

- [.NET Framework 4.8.1 Web Installer](#)
- [.NET Framework 4.8.1 Offline Installer](#)

.NET Framework 4.8 can be installed on Windows 11, Windows 10 version 21H2, Windows 10 version 21H1, Windows 10 version 20H2, and the corresponding server platforms starting with Windows Server 2022. You can install .NET Framework 4.8.1 by using either the web installer or the offline installer. The recommended way for most users is to use the web installer.

You can target .NET Framework 4.8.1 in Visual Studio 2022 17.3 or later by installing the [.NET Framework 4.8.1 Developer Pack](#).

What's new in .NET Framework 4.8.1

.NET Framework 4.8.1 introduces new features in the following areas:

- [Native support for Arm64](#)
- [WCAG2.1 compliant accessible tooltips](#)
- [Windows Forms – Accessibility Improvements](#)

Improved accessibility, which allows an application to provide an appropriate experience for users of Assistive Technology, is a major focus of .NET Framework 4.8.1. For information on accessibility improvements in .NET Framework 4.8.1, see [What's new in accessibility in .NET Framework](#).

.NET Framework 4.8.1 adds native Arm64 support to the .NET Framework family. So, your investments in the vast ecosystem of .NET Framework apps and libraries can now leverage the benefits of running workloads natively on Arm64—namely better performance when compared to running x64 code emulated on Arm64.

Microsoft has a commitment to providing products and platforms that are [accessible to everyone](#). .NET Framework 4.8.1 offers two Windows UI development platforms, both of which provide developers with the support necessary to create accessible applications. Over the past several releases, both Windows Forms and WPF have added

new features and fixed numerous reliability issues related to accessibility. You can read more about the details of what was fixed or added in each release by visiting [What's new in accessibility in .NET Framework](#).

In this release, both Windows Forms and WPF have made improvements to the handling of tooltips to make them more accessible. In both cases, tooltips now comply with the guidelines set forth in the [WCAG2.1 content on Hover or Focus](#) guidance. The requirements for tooltips are:

- Tooltips must display either via mouse hover or by keyboard navigation to the control.
- Tooltips should be dismissable. That is, a simple keyboard command like `Esc` should dismiss the tooltip.
- Tooltips should be hoverable. Users should be able to place their mouse cursor over the tooltip. This enables scenarios like using magnifier to be able to read the tooltip for low-vision users.
- Tooltips should be persistent. Tooltips should not automatically disappear after a certain amount of time has elapsed. Rather, tooltips should be dismissed by the user moving their mouse to another control or by a keyboard command.

In Windows Forms, this support is only available on Windows 11 or later operating systems. Windows Forms is a thin managed wrapper around the Windows API, and the new tooltip behavior only became available in Windows 11. WPF has no operating system version dependencies for its accessible tooltips.

WPF had implemented most of the requirements for WCAG2.1 compliant tooltips in .NET Framework 4.8. In this release, WPF improved the experience by ensuring that a tooltip in the current window can easily be dismissed by using the `Esc` key, the `Ctrl` key (by itself), or by the combination `Ctrl` + `Shift` + `F10`. The scope of the escape key was reduced in this release to apply only to the current window. Previously it applied to any open tooltip in the application.

Windows Forms was the first Windows UI stack created for .NET Framework. As such, it was originally created to utilize legacy accessibility technology, which doesn't meet current accessibility requirements. In this release, Windows Forms has addressed a number of issues. For a complete list of the accessibility related changes, visit [What's new in accessibility in .NET Framework](#).

The highlights of Windows Forms improvements in .NET Framework 4.8.1 are:

- **Text pattern support**—Windows Forms added support for the UIA Text Pattern. This pattern enables assistive technology to traverse the content of a TextBox or similar text-based control letter by letter. It enables text to be selected within the control

and changed, and new text to be inserted at the cursor. Windows Forms added this support for TextBox, DataGridView cells, ComboBox controls, and more.

- Address contrast issues— In several controls, Windows Forms has changed the contrast ratio of selection rectangles to be darker and more visible.
- Fixed several DataGridView issues:
 - The scrollbar names have been updated to be consistent.
 - Narrator is now able to focus on empty DataGridView cells.
 - Developers are able to set the localized control type property for Custom DataGridView cells.
 - The link color for DataGridViewLink cells has been updated to have better contrast with the background.

Introducing .NET Framework 4.8

.NET Framework 4.8 builds on previous versions of .NET Framework 4.x by adding many new fixes and several new features while remaining a very stable product.

Download and install .NET Framework 4.8

You can download .NET Framework 4.8 from the following locations:

- [.NET Framework 4.8 Web Installer ↗](#)
- [.NET Framework 4.8 Offline Installer ↗](#)

.NET Framework 4.8 can be installed on Windows 10, Windows 8.1, Windows 7 SP1, and the corresponding server platforms starting with Windows Server 2008 R2 SP1. You can install .NET Framework 4.8 by using either the web installer or the offline installer. The recommended way for most users is to use the web installer.

You can target .NET Framework 4.8 in Visual Studio 2012 or later by installing the [.NET Framework 4.8 Developer Pack ↗](#).

What's new in .NET Framework 4.8

.NET Framework 4.8 introduces new features in the following areas:

- [Base classes](#)
- [Windows Communication Foundation \(WCF\)](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Common language runtime](#)

Improved accessibility, which allows an application to provide an appropriate experience for users of Assistive Technology, continues to be a major focus of .NET Framework 4.8. For information on accessibility improvements in .NET Framework 4.8, see [What's new in accessibility in .NET Framework](#).

Base classes

Reduced FIPS impact on Cryptography. In previous versions of .NET Framework, managed cryptographic provider classes such as [SHA256Managed](#) throw a [CryptographicException](#) when the system cryptographic libraries are configured in "FIPS mode". These exceptions are thrown because the managed versions of the cryptographic provider classes, unlike the system cryptographic libraries, have not undergone FIPS (Federal Information Processing Standards) 140-2 certification. Because few developers have their development machines in FIPS mode, the exceptions are commonly thrown in production systems.

By default in applications that target .NET Framework 4.8, the following managed cryptography classes no longer throw a [CryptographicException](#) in this case:

- [MD5Cng](#)
- [MD5CryptoServiceProvider](#)
- [RC2CryptoServiceProvider](#)
- [RijndaelManaged](#)
- [RIPEMD160Managed](#)
- [SHA256Managed](#)

Instead, these classes redirect cryptographic operations to a system cryptography library. This change effectively removes a potentially confusing difference between developer environments and production environments and makes native components and managed components operate under the same cryptographic policy. Applications that depend on these exceptions can restore the previous behavior by setting the AppContext switch `Switch.System.Security.Cryptography.UseLegacyFipsThrow` to `true`. For more information, see [Managed cryptography classes do not throw a CryptographicException in FIPS mode](#).

Use of updated version of ZLib

Starting with .NET Framework 4.5, the `clrcompression.dll` assembly uses [ZLib](#), a native external library for data compression, in order to provide an implementation for the deflate algorithm. The .NET Framework 4.8 version of `clrcompression.dll` is updated to use ZLib Version 1.2.11, which includes several key improvements and fixes.

Windows Communication Foundation (WCF)

Introduction of ServiceHealthBehavior

Health endpoints are widely used by orchestration tools to manage services based on their health status. Health checks can also be used by monitoring tools to track and provide notifications about the availability and performance of a service.

ServiceHealthBehavior is a WCF service behavior that extends [IServiceBehavior](#). When added to the [ServiceDescription.Behaviors](#) collection, a service behavior does the following:

- Returns service health status with HTTP response codes. You can specify in a query string the HTTP status code for a HTTP/GET health probe request.
- Publishes information about service health. Service-specific details, including service state, throttle counts, and capacity can be displayed by using an HTTP/GET request with the `?health` query string. Ease of access to such information is important when troubleshooting a misbehaving WCF service.

There are two ways to expose the health endpoint and publish WCF service health information:

- Through code. For example:

C#

```
ServiceHost host = new ServiceHost(typeof(Service1),
    new Uri("http://contoso:81/Service1"));
ServiceHealthBehavior healthBehavior =
    host.Description.Behaviors.Find<ServiceHealthBehavior>();
healthBehavior ??= new ServiceHealthBehavior();
host.Description.Behaviors.Add(healthBehavior);
```

- By using a configuration file. For example:

XML

```
<behaviors>
  <serviceBehaviors>
    <behavior name="DefaultBehavior">
      <serviceHealth httpsGetEnabled="true"/>
    </behavior>
  </serviceBehaviors>
</behaviors>
```

A service's health status can be queried by using query parameters such as `OnServiceFailure`, `OnDispatcherFailure`, `OnListenerFailure`, `OnThrottlePercentExceeded`), and an HTTP response code can be specified for each query parameter. If the HTTP response code is omitted for a query parameter, a 503 HTTP response code is used by default. For example:

- OnServiceFailure: `https://contoso:81/Service1?health&OnServiceFailure=450`

A 450 HTTP response status code is returned when `ServiceHost.State` is greater than `CommunicationState.Opened`.

Query parameters and examples:

- OnDispatcherFailure: `https://contoso:81/Service1?`
`health&OnDispatcherFailure=455`

A 455 HTTP response status code is returned when the state of any of the channel dispatchers is greater than `CommunicationState.Opened`.

- OnListenerFailure: `https://contoso:81/Service1?health&OnListenerFailure=465`

A 465 HTTP response status code is returned when the state of any of the channel listeners is greater than `CommunicationState.Opened`.

- OnThrottlePercentExceeded: `https://contoso:81/Service1?`
`health&OnThrottlePercentExceeded= 70:350,95:500`

Specifies the percentage {1 – 100} that triggers the response and its HTTP response code {200 – 599}. In this example:

- If the percentage is greater than 95, a 500 HTTP response code is returned.
- If the percentage is between 70 and 95, 350 is returned.
- Otherwise, 200 is returned.

The service health status can be displayed either in HTML by specifying a query string like `https://contoso:81/Service1?health` or in XML by specifying a query string like `https://contoso:81/Service1?health&Xml`. A query string like `https://contoso:81/Service1?health&NoContent` returns an empty HTML page.

Windows Presentation Foundation (WPF)

High DPI enhancements

In .NET Framework 4.8, WPF adds support for Per-Monitor V2 DPI Awareness and Mixed-Mode DPI scaling. See [High DPI Desktop Application Development on Windows](#) for additional information about high DPI development.

.NET Framework 4.8 improves support for hosted HWNDs and Windows Forms interoperation in High-DPI WPF applications on platforms that support Mixed-Mode DPI scaling (starting with Windows 10 April 2018 Update). When hosted HWNDs or Windows Forms controls are created as Mixed-Mode DPI-scaled windows by calling [SetThreadDpiHostingBehavior](#) and [SetThreadDpiAwarenessContext](#), they can be hosted in a Per-Monitor V2 WPF application and are sized and scaled appropriately. Such hosted content is not rendered at the native DPI; instead, the operating system scales the hosted content to the appropriate size. The support for Per-Monitor v2 DPI awareness mode also allows WPF controls to be hosted (that is, parented) in a native window in a high-DPI application.

To enable support for Mixed-Mode High DPI scaling, you can set the following [AppContext](#) switches the application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides value =
"Switch.System.Windows.DoNotScaleForDpiChanges=false;
Switch.System.Windows.DoNotUsePresentationDpiCapabilityTier2OrGreater=false"
/>
</runtime>
```

Common language runtime

The runtime in .NET Framework 4.8 includes the following changes and improvements:

Improvements to the JIT compiler. The Just-in-time (JIT) compiler in .NET Framework 4.8 is based on the JIT compiler in .NET Core 2.1. Many of the optimizations and all of the bug fixes made to the .NET Core 2.1 JIT compiler are included in the .NET Framework 4.8 JIT compiler.

NGEN improvements. The runtime has improved its memory management for [Native Image Generator](#) (NGEN) images so that data mapped from NGEN images are not memory-resident. This reduces the surface area available to attacks that attempt to execute arbitrary code by modifying memory that will be executed.

Antimalware scanning for all assemblies. In previous versions of .NET Framework, the runtime scans all assemblies loaded from disk using either Windows Defender or third-party antimalware software. However, assemblies loaded from other sources, such as by

the `Assembly.Load(Byte[])` method, are not scanned and can potentially contain undetected malware. Starting with .NET Framework 4.8 running on Windows 10, the runtime triggers a scan by antimalware solutions that implement the [Antimalware Scan Interface \(AMSI\)](#).

What's new in .NET Framework 4.7.2

.NET Framework 4.7.2 includes new features in the following areas:

- [Base classes](#)
- [ASP.NET](#)
- [Networking](#)
- [SQL](#)
- [WPF](#)
- [ClickOnce](#)

A continuing focus in .NET Framework 4.7.2 is improved accessibility, which allows an application to provide an appropriate experience for users of Assistive Technology. For information on accessibility improvements in .NET Framework 4.7.2, see [What's new in accessibility in .NET Framework](#).

Base classes

.NET Framework 4.7.2 features a large number of cryptographic enhancements, better decompression support for ZIP archives, and additional collection APIs.

New overloads of RSA.Create and DSA.Create

The [DSA.Create\(DSAParameters\)](#) and [RSA.Create\(RSAParameters\)](#) methods let you supply key parameters when instantiating a new [DSA](#) or [RSA](#) key. They allow you to replace code like the following:

```
C#  
  
// Before .NET Framework 4.7.2  
using (RSA rsa = RSA.Create())  
{  
    rsa.ImportParameters(rsaParameters);  
    // Other code to execute using the RSA instance.  
}
```

with code like this:

```
C#
```

```
// Starting with .NET Framework 4.7.2
using (RSA rsa = RSA.Create(rsaParameters))
{
    // Other code to execute using the rsa instance.
}
```

The [DSA.Create\(Int32\)](#) and [RSA.Create\(Int32\)](#) methods let you generate new [DSA](#) or [RSA](#) keys with a specific key size. For example:

C#

```
using (DSA dsa = DSA.Create(2048))
{
    // Other code to execute using the dsa instance.
}
```

Rfc2898DeriveBytes constructors accept a hash algorithm name

The [Rfc2898DeriveBytes](#) class has three new constructors with a [HashAlgorithmName](#) parameter that identifies the HMAC algorithm to use when deriving keys. Instead of using SHA-1, developers should use a SHA-2-based HMAC like SHA-256, as shown in the following example:

C#

```
private static byte[] DeriveKey(string password, out int iterations, out
byte[] salt,
                                out HashAlgorithmName algorithm)
{
    iterations = 100000;
    algorithm = HashAlgorithmName.SHA256;

    const int SaltSize = 32;
    const int DerivedValueSize = 32;

    using (Rfc2898DeriveBytes pbkdf2 = new Rfc2898DeriveBytes(password,
SaltSize,
                                                               iterations,
algorithm))
    {
        salt = pbkdf2.Salt;
        return pbkdf2.GetBytes(DerivedValueSize);
    }
}
```

Support for ephemeral keys

PFX import can optionally load private keys directly from memory, bypassing the hard drive. When the new [X509KeyStorageFlags.EphemeralKeySet](#) flag is specified in an [X509Certificate2](#) constructor or one of the overloads of the [X509Certificate2.Import](#) method, the private keys will be loaded as ephemeral keys. This prevents the keys from being visible on the disk. However:

- Since the keys are not persisted to disk, certificates loaded with this flag are not good candidates to add to an X509Store.
- Keys loaded in this manner are almost always loaded via Windows CNG. Therefore, callers must access the private key by calling extension methods, such as `cert.GetRSAPrivateKey()`. The [X509Certificate2.PrivateKey](#) property does not function.
- Since the legacy [X509Certificate2.PrivateKey](#) property does not work with certificates, developers should perform rigorous testing before switching to ephemeral keys.

Programmatic creation of PKCS#10 certification signing requests and X.509 public key certificates

Starting with .NET Framework 4.7.2, workloads can generate certificate signing requests (CSRs), which allows certificate request generation to be staged into existing tooling. This is frequently useful in test scenarios.

For more information and code examples, see "Programmatic creation of PKCS#10 certification signing requests and X.509 public key certificates" in the [.NET Blog ↗](#).

New SignerInfo members

Starting with .NET Framework 4.7.2, the [SignerInfo](#) class exposes more information about the signature. You can retrieve the value of the [System.Security.Cryptography.Pkcs.SignerInfo.SignatureAlgorithm](#) property to determine the signature algorithm used by the signer. [SignerInfo.GetSignature](#) can be called to get a copy of the cryptographic signature for this signer.

Leaving a wrapped stream open after CryptoStream is disposed

Starting with .NET Framework 4.7.2, the [CryptoStream](#) class has an additional constructor that allows [Dispose](#) to not close the wrapped stream. To leave the wrapped stream open after the [CryptoStream](#) instance is disposed, call the new [CryptoStream](#) constructor as follows:

C#

```
var cStream = new CryptoStream(stream, transform, mode, leaveOpen: true);
```

Decompression changes in DeflateStream

Starting with .NET Framework 4.7.2, the implementation of decompression operations in the [DeflateStream](#) class has changed to use native Windows APIs by default. Typically, this results in a substantial performance improvement.

Support for decompression by using Windows APIs is enabled by default for applications that target .NET Framework 4.7.2. Applications that target earlier versions of .NET Framework but are running under .NET Framework 4.7.2 can opt into this behavior by adding the following [AppContext switch](#) to the application configuration file:

XML

```
<AppContextSwitchOverrides  
value="Switch.System.IO.Compression.DoNotUseNativeZipLibraryForDecompression  
=false" />
```

Additional collection APIs

.NET Framework 4.7.2 adds a number of new APIs to the [SortedSet<T>](#) and [HashSet<T>](#) types. These include:

- `TryGetValue` methods, which extend the try pattern used in other collection types to these two types. The methods are:
 - `public bool HashSet<T>.TryGetValue(T equalValue, out T actualValue)`
 - `public bool SortedSet<T>.TryGetValue(T equalValue, out T actualValue)`
- `Enumerable.To*` extension methods, which convert a collection to a [HashSet<T>](#):
 - `public static HashSet<TSource> ToHashSet<TSource>(this IEnumerable<TSource> source)`
 - `public static HashSet<TSource> ToHashSet<TSource>(this IEnumerable<TSource> source, IEqualityComparer<TSource> comparer)`
- New [HashSet<T>](#) constructors that let you set the collection's capacity, which yields a performance benefit when you know the size of the [HashSet<T>](#) in advance:
 - `public HashSet(int capacity)`
 - `public HashSet(int capacity, IEqualityComparer<T> comparer)`

The [ConcurrentDictionary<TKey,TValue>](#) class includes new overloads of the [AddOrUpdate](#) and [GetOrAdd](#) methods to retrieve a value from the dictionary or to add

it if it is not found, and to add a value to the dictionary or to update it if it already exists.

C#

```
public TValue AddOrUpdate<TArg>(TKey key, Func<TKey, TArg, TValue>
addValueFactory, Func<TKey, TValue, TArg, TValue> updateValueFactory, TArg
factoryArgument)

public TValue GetOrAdd<TArg>(TKey key, Func<TKey, TArg, TValue>
valueFactory, TArg factoryArgument)
```

ASP.NET

Support for dependency injection in Web Forms

Dependency injection (DI) decouples objects and their dependencies so that an object's code no longer needs to be changed just because a dependency has changed. When developing ASP.NET applications that target .NET Framework 4.7.2, you can:

- Use setter-based, interface-based, and constructor-based injection in [handlers and modules](#), [Page instances](#), and [user controls](#) of ASP.NET web application projects.
- Use setter-based and interface-based injection in [handlers and modules](#), [Page instances](#), and [user controls](#) of ASP.NET web site projects.
- Plug in different dependency injection frameworks.

Support for same-site cookies

[SameSite](#) prevents a browser from sending a cookie along with a cross-site request. .NET Framework 4.7.2 adds a [HttpCookie.SameSite](#) property whose value is a [System.Web.SameSiteMode](#) enumeration member. If its value is [SameSiteMode.Strict](#) or [SameSiteMode.Lax](#), ASP.NET adds the `SameSite` attribute to the set-cookie header. SameSite support applies to [HttpCookie](#) objects, as well as to [FormsAuthentication](#) and [System.Web.SessionState](#) cookies.

You can set SameSite for an [HttpCookie](#) object as follows:

C#

```
var c = new HttpCookie("secureCookie", "same origin");
c.SameSite = SameSiteMode.Lax;
```

You can also configure SameSite cookies at the application level by modifying the web.config file:

XML

```
<system.web>
  <httpCookies sameSite="Strict" />
</system.web>
```

You can add SameSite for [FormsAuthentication](#) and [System.Web.SessionState](#) cookies by modifying the web config file:

XML

```
<system.web>
  <authentication mode="Forms">
    <forms cookieSameSite="Lax">
      <!-- ... -->
    </forms>
  </authentication>
  <sessionState cookieSameSite="Lax"></sessionState>
</system.web>
```

Networking

Implementation of HttpClientHandler properties

.NET Framework 4.7.1 added eight properties to the [System.Net.Http.HttpClientHandler](#) class. However, two threw a [PlatformNotSupportedException](#). .NET Framework 4.7.2 now provides an implementation for these properties. The properties are:

- [CheckCertificateRevocationList](#)
- [SslProtocols](#)

SQLClient

Support for Azure Active Directory Universal Authentication and Multifactor authentication

Growing compliance and security demands require that many customers use multifactor authentication (MFA). In addition, current best practices discourage including user passwords directly in connection strings. To support these changes, .NET Framework 4.7.2 extends [SQLClient connection strings](#) by adding a new value, "Active Directory Interactive", for the existing "Authentication" keyword to support MFA and [Azure AD Authentication](#). The new interactive method supports native and federated Azure AD users as well as Azure AD guest users. When this method is used, the MFA

authentication imposed by Azure AD is supported for SQL databases. In addition, the authentication process requests a user password to adhere to security best practices.

In previous versions of .NET Framework, SQL connectivity supported only the [SqlAuthenticationMethod.ActiveDirectoryPassword](#) and [SqlAuthenticationMethod.ActiveDirectoryIntegrated](#) options. Both of these are part of the non-interactive [ADAL protocol](#), which does not support MFA. With the new [SqlAuthenticationMethod.ActiveDirectoryInteractive](#) option, SQL connectivity supports MFA as well as existing authentication methods (password and integrated authentication), which allows users to enter user passwords interactively without persisting passwords in the connection string.

For more information and an example, see "SQL -- Azure AD Universal and Multifactor Authentication Support" in the [.NET Blog ↗](#).

Support for Always Encrypted version 2

.NET Framework 4.7.2 adds supports for enclave-based Always Encrypted. The original version of Always Encrypted is a client-side encryption technology in which encryption keys never leave the client. In enclave-based Always Encrypted, the client can optionally send the encryption keys to a secure enclave, which is a secure computational entity that can be considered part of SQL Server but that SQL Server code cannot tamper with. To support enclave-based Always Encrypted, .NET Framework 4.7.2 adds the following types and members to the [System.Data.SqlClient](#) namespace:

- [SqlConnectionStringBuilder.EnclaveAttestationUrl](#), which specifies the Uri for enclave-based Always Encrypted.
- [SqlColumnEncryptionEnclaveProvider](#), which is an abstract class from which all enclave providers are derived.
- [SqlEnclaveSession](#), which encapsulates the state for a given enclave session.
- [SqlEnclaveAttestationParameters](#), which provides the attestation parameters used by SQL Server to get information required to execute a particular Attestation Protocol.

The application configuration file then specifies a concrete implementation of the abstract [System.Data.SqlClient.SqlColumnEncryptionEnclaveProvider](#) class that provides the functionality for the enclave provider. For example:

XML

```
<configuration>
  <configSections>
```

```

<section name="SqlColumnEncryptionEnclaveProviders"
type="System.Data.SqlClient.SqlColumnEncryptionEnclaveProviderConfigurationSection, System.Data, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
</configSections>
<SqlColumnEncryptionEnclaveProviders>
  <providers>
    <add name="Azure"
type="Microsoft.SqlServer.Management.AlwaysEncrypted.AzureEnclaveProvider, MyApp"/>
    <add name="HGS"
type="Microsoft.SqlServer.Management.AlwaysEncrypted.HGSEnclaveProvider, MyAp
p" />
  </providers>
</SqlColumnEncryptionEnclaveProviders >
</configuration>

```

The basic flow of enclave-based Always Encrypted is:

1. The user creates an AlwaysEncrypted connection to SQL Server that supported enclave-based Always Encrypted. The driver contacts the attestation service to ensure that it is connecting to right enclave.
2. Once the enclave has been attested, the driver establishes a secure channel with the secure enclave hosted on SQL Server.
3. The driver shares encryption keys authorized by the client with the secure enclave for the duration of the SQL connection.

Windows Presentation Foundation

Finding ResourceDictionaries by Source

Starting with .NET Framework 4.7.2, a diagnostic assistant can locate the [ResourceDictionaries](#) that have been created from a given source Uri. (This feature is for use by diagnostic assistants, not by production applications.) A diagnostic assistant such as Visual Studio's "Edit-and-Continue" facility lets its user edit a ResourceDictionary with the intent that the changes be applied to the running application. One step in achieving this is finding all the ResourceDictionaries that the running application has created from the dictionary that's being edited. For example, an application can declare a ResourceDictionary whose content is copied from a given source URI:

XML

```
<ResourceDictionary Source="MyRD.xaml" />
```

A diagnostic assistant that edits the original markup in *MyRD.xaml* can use the new feature to locate the dictionary. The feature is implemented by a new static method, [ResourceDictionaryDiagnostics.GetResourceDictionariesForSource](#). The diagnostic assistant calls the new method using an absolute Uri that identifies the original markup, as illustrated by the following code:

C#

```
IEnumerable<ResourceDictionary> dictionaries =  
    ResourceDictionaryDiagnostics.GetResourceDictionariesForSource(new  
Uri\("pack://application:,,,/MyApp;component/MyRD.xaml"\)\);
```

The method returns an empty enumerable unless [VisualDiagnostics](#) is enabled and the [ENABLE_XAML_DIAGNOSTICS_SOURCE_INFO](#) environment variable is set.

Finding ResourceDictionary owners

Starting with .NET Framework 4.7.2, a diagnostic assistant can locate the owners of a given [ResourceDictionary](#). (The feature is for use by diagnostic assistants and not by production applications.) Whenever a change is made to a [ResourceDictionary](#), WPF automatically finds all [DynamicResource](#) references that might be affected by the change.

A diagnostic assistant such as Visual Studio's "Edit-and-Continue" facility may want to extend this to handle [StaticResource](#) references. The first step in this process is to find the owners of the dictionary; that is, to find all the objects whose [Resources](#) property refers to the dictionary (either directly, or indirectly via the [ResourceDictionary.MergedDictionaries](#) property). Three new static methods implemented on the [System.Windows.Diagnostics.ResourceDictionaryDiagnostics](#) class, one for each of the base types that has a [Resources](#) property, support this step:

- [public static IEnumerable<FrameworkElement>
GetFrameworkElementOwners\(ResourceDictionary dictionary\);](#)
- [public static IEnumerable<FrameworkContentElement>
GetFrameworkContentElementOwners\(ResourceDictionary dictionary\);](#)
- [public static IEnumerable<Application> GetApplicationOwners\(ResourceDictionary
dictionary\);](#)

These methods return an empty enumerable unless [VisualDiagnostics](#) is enabled and the [ENABLE_XAML_DIAGNOSTICS_SOURCE_INFO](#) environment variable is set.

Finding StaticResource references

A diagnostic assistant can now receive a notification whenever a [StaticResource](#) reference is resolved. (The feature is for use by diagnostic assistants, not by production applications.) A diagnostic assistant such as Visual Studio's "Edit-and-Continue" facility may want to update all uses of a resource when its value in a [ResourceDictionary](#) changes. WPF does this automatically for [DynamicResource](#) references, but it intentionally does not do so for [StaticResource](#) references. Starting with .NET Framework 4.7.2, the diagnostic assistant can use these notifications to locate those uses of the static resource.

The notification is implemented by the new [ResourceDictionaryDiagnostics.StaticResourceResolved](#) event:

C#

```
public static event EventHandler<StaticResourceResolvedEventArgs>
StaticResourceResolved;
```

This event is raised whenever the runtime resolves a [StaticResource](#) reference. The [StaticResourceResolvedEventArgs](#) arguments describe the resolution, and indicate the object and property that host the [StaticResource](#) reference and the [ResourceDictionary](#) and key used for the resolution:

C#

```
public class StaticResourceResolvedEventArgs : EventArgs
{
    public Object TargetObject { get; }

    public Object TargetProperty { get; }

    public ResourceDictionary ResourceDictionary { get; }

    public object ResourceKey { get; }
}
```

The event is not raised (and its `add` accessor is ignored) unless [VisualDiagnostics](#) is enabled and the [ENABLE_XAML_DIAGNOSTICS_SOURCE_INFO](#) environment variable is set.

ClickOnce

HDPI-aware applications for Windows Forms, Windows Presentation Foundation (WPF), and Visual Studio Tools for Office (VSTO) can all be deployed by using ClickOnce. If the

following entry is found in the application manifest, deployment will succeed under .NET Framework 4.7.2:

XML

```
<windowsSettings>
  <dpiAware>
    <xmlNamespace value="http://schemas.microsoft.com/SMI/2005/WindowsSettings" />true
  </dpiAware>
</windowsSettings>
```

For Windows Forms application, the previous workaround of setting DPI awareness in the application configuration file rather than the application manifest is no longer necessary for ClickOnce deployment to succeed.

What's new in .NET Framework 4.7.1

.NET Framework 4.7.1 includes new features in the following areas:

- [Base classes](#)
- [Common language runtime \(CLR\)](#)
- [Networking](#)
- [ASP.NET](#)

In addition, a major focus in .NET Framework 4.7.1 is improved accessibility, which allows an application to provide an appropriate experience for users of Assistive Technology. For information on accessibility improvements in .NET Framework 4.7.1, see [What's new in accessibility in .NET Framework](#).

Base classes

Support for .NET Standard 2.0

[.NET Standard](#) defines a set of APIs that must be available on each .NET implementation that supports that version of the standard. .NET Framework 4.7.1 fully supports .NET Standard 2.0 and adds [about 200 APIs](#) that are defined in .NET Standard 2.0 and are missing from .NET Framework 4.6.1, 4.6.2, and 4.7. (Note that these versions of .NET Framework support .NET Standard 2.0 only if additional .NET Standard support files are also deployed on the target system.) For more information, see "BCL - .NET Standard 2.0 Support" in the [.NET Framework 4.7.1 Runtime and Compiler Features](#) blog post.

Support for configuration builders

Configuration builders allow developers to inject and build configuration settings for applications dynamically at run time. Custom configuration builders can be used to modify existing data in a configuration section or to build a configuration section entirely from scratch. Without configuration builders, .config files are static, and their settings are defined some time before an application is launched.

To create a custom configuration builder, you derive your builder from the abstract [ConfigurationBuilder](#) class and override its [ConfigurationBuilder.ProcessConfigurationSection](#) and [ConfigurationBuilder.ProcessRawXml](#). You also define your builders in your .config file. For more information, see the "Configuration Builders" section in the [.NET Framework 4.7.1 ASP.NET and Configuration Features](#) blog post.

Run-time feature detection

The [System.Runtime.CompilerServices.RuntimeFeature](#) class provides a mechanism for determine whether a predefined feature is supported on a given .NET implementation at compile time or run time. At compile time, a compiler can check whether a specified field exists to determine whether the feature is supported; if so, it can emit code that takes advantage of that feature. At run time, an application can call the [RuntimeFeature.IsSupported](#) method before emitting code at run time. For more information, see [Add helper method to describe features supported by the runtime](#).

Value tuple types are serializable

Starting with .NET Framework 4.7.1, [System.ValueTuple](#) and its associated generic types are marked as [Serializable](#), which allows binary serialization. This should make migrating Tuple types, such as [Tuple<T1,T2,T3>](#) and [Tuple<T1,T2,T3,T4>](#), to value tuple types easier. For more information, see "Compiler -- ValueTuple is Serializable" in the [.NET Framework 4.7.1 Runtime and Compiler Features](#) blog post.

Support for read-only references

.NET Framework 4.7.1 adds the [System.Runtime.CompilerServices.IsReadOnlyAttribute](#). This attribute is used by language compilers to mark members that have read-only ref return types or parameters. For more information, see "Compiler -- Support for ReadOnlyReferences" in the [.NET Framework 4.7.1 Runtime and Compiler Features](#) blog post. For information on ref return values, see [Ref return values](#) and [ref locals](#) and [Ref return values \(Visual Basic\)](#).

Common language runtime (CLR)

Garbage collection performance improvements

Changes to garbage collection (GC) in .NET Framework 4.7.1 improve overall performance, especially for large object heap (LOH) allocations. In .NET Framework 4.7.1, separate locks are used for small object heap (SOH) and LOH allocations, which allows LOH allocations to occur when background GC is sweeping the SOH. As a result, applications that make a large number of LOH allocations should see a reduction in allocation lock contention and improved performance. For more information, see the "Runtime -- GC Performance Improvements" section in the [.NET Framework 4.7.1 Runtime and Compiler Features](#) blog post.

Networking

SHA-2 support for Message.HashAlgorithm

In .NET Framework 4.7 and earlier versions, the `Message.HashAlgorithm` property supported values of `HashAlgorithm.Md5` and `HashAlgorithm.Sha` only. Starting with .NET Framework 4.7.1, `HashAlgorithm.Sha256`, `HashAlgorithm.Sha384`, and `HashAlgorithm.Sha512` are also supported. Whether this value is actually used depends on MSMQ, since the `Message` instance itself does no hashing but simply passes on values to MSMQ. For more information, see the "SHA-2 support for `Message.HashAlgorithm`" section in the [.NET Framework 4.7.1 ASP.NET and Configuration features](#) blog post.

ASP.NET

Execution steps in ASP.NET applications

ASP.NET processes requests in a predefined pipeline that includes 23 events. ASP.NET executes each event handler as an execution step. In versions of ASP.NET up to .NET Framework 4.7, ASP.NET can't flow the execution context due to switching between native and managed threads. Instead, ASP.NET selectively flows only the `HttpContext`. Starting with .NET Framework 4.7.1, the `HttpApplication.OnExecuteRequestStep(Action<HttpContextBase,Action>)` method also allows modules to restore ambient data. This feature is targeted at libraries concerned with tracing, profiling, diagnostics, or transactions, for example, that care about the execution flow of the application. For more information, see the "ASP.NET Execution Step Feature" in the [.NET Framework 4.7.1 ASP.NET and Configuration Features](#) blog post.

ASP.NET HttpCookie parsing

.NET Framework 4.7.1 includes a new method, `HttpCookie.TryParse`, that provides a standardized way to create an `HttpCookie` object from a string and accurately assign

cookie values such as expiration date and path. For more information, see "ASP.NET HttpCookie parsing" in the [.NET Framework 4.7.1 ASP.NET and Configuration Features](#) blog post.

SHA-2 hash options for ASP.NET forms authentication credentials

In .NET Framework 4.7 and earlier versions, ASP.NET allowed developers to store user credentials with hashed passwords in configuration files using either MD5 or SHA1. Starting with .NET Framework 4.7.1, ASP.NET also supports new secure SHA-2 hash options such as SHA256, SHA384, and SHA512. SHA1 remains the default, and a non-default hash algorithm can be defined in the web configuration file. For example:

XML

```
<system.web>
  <authentication mode="Forms">
    <forms loginUrl("~/login.aspx")>
      <credentials passwordFormat="SHA512">
        <user name="jdoe"
password="6D003E98EA1C7F04ABF8FCB375388907B7F3EE06F278DB966BE960E7CBB103DF3
0CA6D61F7E7FD981B2E4E3A64D43C836A4BEDCA165C33B163E6BCDC538A664" />
      </credentials>
    </forms>
  </authentication>
</system.web>
```

What's new in .NET Framework 4.7

.NET Framework 4.7 includes new features in the following areas:

- [Base classes](#)
- [Networking](#)
- [ASP.NET](#)
- [Windows Communication Foundation \(WCF\)](#)
- [Windows Forms](#)
- [Windows Presentation Foundation \(WPF\)](#)

For a list of new APIs added to .NET Framework 4.7, see [.NET Framework 4.7 API Changes](#) on GitHub. For a list of feature improvements and bug fixes in .NET Framework 4.7, see [.NET Framework 4.7 List of Changes](#) on GitHub. For more information, see [Announcing .NET Framework 4.7](#) in the .NET blog.

Base classes

.NET Framework 4.7 improves serialization by the [DataContractJsonSerializer](#):

Enhanced functionality with Elliptic Curve Cryptography (ECC)*

In .NET Framework 4.7, `ImportParameters(ECParameters)` methods were added to the `ECDsa` and `ECDiffieHellman` classes to allow for an object to represent an already-established key. An `ExportParameters(Boolean)` method was also added for exporting the key using explicit curve parameters.

.NET Framework 4.7 also adds support for additional curves (including the Brainpool curve suite), and has added predefined definitions for ease-of-creation through the new `Create` and `Create` factory methods.

You can see an [example of .NET Framework 4.7 cryptography improvements ↗](#) on GitHub.

Better support for control characters by the DataContractJsonSerializer

In .NET Framework 4.7, the `DataContractJsonSerializer` class serializes control characters in conformity with the ECMAScript 6 standard. This behavior is enabled by default for applications that target .NET Framework 4.7, and is an opt-in feature for applications that are running under .NET Framework 4.7 but target a previous version of .NET Framework. For more information, see the [Application compatibility](#) section.

Networking

.NET Framework 4.7 adds the following network-related feature:

Default operating system support for TLS protocols*

The TLS stack, which is used by `System.Net.Security.SslStream` and up-stack components such as HTTP, FTP, and SMTP, allows developers to use the default TLS protocols supported by the operating system. Developers need no longer hard-code a TLS version.

ASP.NET

In .NET Framework 4.7, ASP.NET includes the following new features:

Object Cache Extensibility

Starting with .NET Framework 4.7, ASP.NET adds a new set of APIs that allow developers to replace the default ASP.NET implementations for in-memory object caching and

memory monitoring. Developers can now replace any of the following three components if the ASP.NET implementation is not adequate:

- **Object Cache Store.** By using the new cache providers configuration section, developers can plug in new implementations of an object cache for an ASP.NET application by using the new **ICacheStoreProvider** interface.
- **Memory monitoring.** The default memory monitor in ASP.NET notifies applications when they are running close to the configured private bytes limit for the process, or when the machine is low on total available physical RAM. When these limits are near, notifications are fired. For some applications, notifications are fired too close to the configured limits to allow for useful reactions. Developers can now write their own memory monitors to replace the default by using the [ApplicationMonitors.MemoryMonitor](#) property.
- **Memory Limit Reactions.** By default, ASP.NET attempts to trim the object cache and periodically call [GC.Collect](#) when the private byte process limit is near. For some applications, the frequency of calls to [GC.Collect](#) or the amount of cache that is trimmed are inefficient. Developers can now replace or supplement the default behavior by subscribing **IObserver** implementations to the application's memory monitor.

Windows Communication Foundation (WCF)

Windows Communication Foundation (WCF) adds the following features and changes:

Ability to configure the default message security settings to TLS 1.1 or TLS 1.2

Starting with .NET Framework 4.7, WCF allows you to configure TLS 1.1 or TLS 1.2 in addition to SSL 3.0 and TLS 1.0 as the default message security protocol. This is an opt-in setting; to enable it, you must add the following entry to your application configuration file:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.ServiceModel.DisableUsingServicePointManagerSecurityPro
          tocols=false;Switch.System.Net.DontEnableSchUseStrongCrypto=false" />
</runtime>
```

Improved reliability of WCF applications and WCF serialization

WCF includes a number of code changes that eliminate race conditions, thereby improving performance and the reliability of serialization options. These include:

- Better support for mixing asynchronous and synchronous code in calls to `SocketConnection.BeginRead` and `SocketConnection.Read`.
- Improved reliability when aborting a connection with `SharedConnectionListener` and `DuplexChannelBinder`.
- Improved reliability of serialization operations when calling the `FormatterServices.GetSerializableMembers(Type)` method.
- Improved reliability when removing a waiter by calling the `ChannelSynchronizer.RemoveWaiter` method.

Windows Forms

In .NET Framework 4.7, Windows Forms improves support for high DPI monitors.

High DPI support

Starting with applications that target .NET Framework 4.7, .NET Framework features high DPI and dynamic DPI support for Windows Forms applications. High DPI support improves the layout and appearance of forms and controls on high DPI monitors. Dynamic DPI changes the layout and appearance of forms and controls when the user changes the DPI or display scale factor of a running application.

High DPI support is an opt-in feature that you configure by defining a `<System.Windows.Forms.ConfigurationSection>` section in your application configuration file. For more information on adding high DPI support and dynamic DPI support to your Windows Forms application, see [High DPI Support in Windows Forms](#).

Windows Presentation Foundation (WPF)

In .NET Framework 4.7, WPF includes the following enhancements:

Support for a touch/stylus stack based on Windows WM_POINTER messages

You now have the option of using a touch/stylus stack based on [WM_POINTER messages](#) instead of the Windows Ink Services Platform (WISP). This is an opt-in feature in .NET Framework. For more information, see the [Application compatibility](#) section.

New implementation for WPF printing APIs

WPF's printing APIs in the `System.Printing.PrintQueue` class call the Windows [Print Document Package API](#) instead of the deprecated [XPS Print API](#). For the impact of this

change on application compatibility, see the [Application compatibility](#) section.

What's new in .NET Framework 4.6.2

.NET Framework 4.6.2 includes new features in the following areas:

- [ASP.NET](#)
- [Character categories](#)
- [Cryptography](#)
- [SqlClient](#)
- [Windows Communication Foundation](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Workflow Foundation \(WF\)](#)
- [ClickOnce](#)
- [Converting Windows Forms and WPF apps to UWP apps](#)
- [Debugging improvements](#)

For a list of new APIs added to .NET Framework 4.6.2, see [.NET Framework 4.6.2 API Changes](#) on GitHub. For a list of feature improvements and bug fixes in .NET Framework 4.6.2, see [.NET Framework 4.6.2 List of Changes](#) on GitHub. For more information, see [Announcing .NET Framework 4.6.2](#) in the .NET blog.

ASP.NET

In .NET Framework 4.6.2, ASP.NET includes the following enhancements:

Improved support for localized error messages in data annotation validators

Data annotation validators enable you to perform validation by adding one or more attributes to a class property. The attribute's `ValidationAttribute(ErrorMessage)` element defines the text of the error message if validation fails. Starting with .NET Framework 4.6.2, ASP.NET makes it easy to localize error messages. Error messages will be localized if:

1. The `ValidationAttribute(ErrorMessage)` is provided in the validation attribute.
2. The resource file is stored in the App_LocalResources folder.

3. The name of the localized resources file has the form

`DataAnnotation.Localization.{name}.resx`, where `name` is a culture name in the format `languageCode - country/regionCode` or `languageCode`.

4. The key name of the resource is the string assigned to the `ValidationAttribute.ErrorMessage` attribute, and its value is the localized error message.

For example, the following data annotation attribute defines the default culture's error message for an invalid rating.

C#

```
public class RatingInfo
{
    [Required(ErrorMessage = "The rating must be between 1 and 10.")]
    [Display(Name = "Your Rating")]
    public int Rating { get; set; }
}
```

You can then create a resource file, `DataAnnotation.Localization.fr.resx`, whose key is the error message string and whose value is the localized error message. The file must be found in the `App.LocalResources` folder. For example, the following is the key and its value in a localized French (fr) language error message:

[+] Expand table

Name	Value
The rating must be between 1 and 10.	La note doit être comprise entre 1 et 10.

In addition, data annotation localization is extensible. Developers can plug in their own string localizer provider by implementing the `IStringLocalizerProvider` interface to store localization string somewhere other than in a resource file.

Async support with session-state store providers

ASP.NET now allows task-returning methods to be used with session-state store providers, thereby allowing ASP.NET apps to get the scalability benefits of async. To support asynchronous operations with session state store providers, ASP.NET includes a new interface, `System.Web.SessionState.ISessionStateModule`, which inherits from `IHttpModule` and allows developers to implement their own session-state module and async session store providers. The interface is defined as follows:

C#

```
public interface ISessionStateModule : IHttpModule {  
    void ReleaseSessionState(HttpContext context);  
    Task ReleaseSessionStateAsync(HttpContext context);  
}
```

In addition, the [SessionStateUtility](#) class includes two new methods, [IsSessionStateReadOnly](#) and [IsSessionStateRequired](#), that can be used to support asynchronous operations.

Async support for output-cache providers

Starting with .NET Framework 4.6.2, task-returning methods can be used with output-cache providers to provide the scalability benefits of async. Providers that implement these methods reduce thread-blocking on a web server and improve the scalability of an ASP.NET service.

The following APIs have been added to support asynchronous output-cache providers:

- The [System.Web.Caching.OutputCacheProviderAsync](#) class, which inherits from [System.Web.Caching.OutputCacheProvider](#) and allows developers to implement an asynchronous output-cache provider.
- The [OutputCacheUtility](#) class, which provides helper methods for configuring the output cache.
- 18 new methods in the [System.Web.HttpCachePolicy](#) class. These include [GetCacheability](#), [GetCacheExtensions](#), [GetETag](#), [GetETagFromFileDependencies](#), [GetMaxAge](#), [GetMaxAge](#), [GetNoStore](#), [GetNoTransforms](#), [GetOmitVaryStar](#), [GetProxyMaxAge](#), [GetRevalidation](#), [GetUtcLastModified](#), [GetVaryByCustom](#), [HasSlidingExpiration](#), and [IsValidUntilExpires](#).
- 2 new methods in the [System.Web.HttpCacheVaryByContentEncodings](#) class: [GetContentEncodings](#) and [SetContentEncodings](#).
- 2 new methods in the [System.Web.HttpCacheVaryByHeaders](#) class: [GetHeaders](#) and [SetHeaders](#).
- 2 new methods in the [System.Web.HttpCacheVaryByParams](#) class: [GetParams](#) and [SetParams](#).
- In the [System.Web.Caching.AggregateCacheDependency](#) class, the [GetFileDependencies](#) method.
- In the [CacheDependency](#), the [GetFileDependencies](#) method.

Character categories

Characters in .NET Framework 4.6.2 are classified based on the [Unicode Standard, Version 8.0.0](#). In .NET Framework 4.6 and .NET Framework 4.6.1, characters were classified based on Unicode 6.3 character categories.

Support for Unicode 8.0 is limited to the classification of characters by the [CharUnicodeInfo](#) class and to types and methods that rely on it. These include the [StringInfo](#) class, the overloaded [Char.GetUnicodeCategory](#) method, and the [character classes](#) recognized by the .NET Framework regular expression engine. Character and string comparison and sorting is unaffected by this change and continues to rely on the underlying operating system or, on Windows 7 systems, on character data provided by .NET Framework.

For changes in character categories from Unicode 6.0 to Unicode 7.0, see [The Unicode Standard, Version 7.0.0](#) at The Unicode Consortium website. For changes from Unicode 7.0 to Unicode 8.0, see [The Unicode Standard, Version 8.0.0](#) at The Unicode Consortium website.

Cryptography

Support for X509 certificates containing FIPS 186-3 DSA

.NET Framework 4.6.2 adds support for DSA (Digital Signature Algorithm) X509 certificates whose keys exceed the FIPS 186-2 1024-bit limit.

In addition to supporting the larger key sizes of FIPS 186-3, .NET Framework 4.6.2 allows computing signatures with the SHA-2 family of hash algorithms (SHA256, SHA384, and SHA512). FIPS 186-3 support is provided by the new [System.Security.Cryptography.DSACng](#) class.

In keeping with recent changes to the [RSA](#) class in .NET Framework 4.6 and the [ECDsa](#) class in .NET Framework 4.6.1, the [DSA](#) abstract base class in .NET Framework 4.6.2 has additional methods to allow callers to use this functionality without casting. You can call the [DSACertificateExtensions.GetDSAPrivateKey](#) extension method to sign data, as the following example shows.

C#

```
public static byte[] SignDataDsaSha384(byte[] data, X509Certificate2 cert)
{
    using (DSA dsa = cert.GetDSAPrivateKey())
    {
        return dsa.SignData(data, HashAlgorithmName.SHA384);
```

```
    }  
}
```

And you can call the [DSACertificateExtensions.GetDSAPublicKey](#) extension method to verify signed data, as the following example shows.

C#

```
public static bool VerifyDataDsaSha384(byte[] data, byte[] signature,  
X509Certificate2 cert)  
{  
    using (DSA dsa = cert.GetDSAPublicKey())  
    {  
        return dsa.VerifyData(data, signature, HashAlgorithmName.SHA384);  
    }  
}
```

Increased clarity for inputs to ECDiffieHellman key derivation routines

.NET Framework 3.5 added support for Elliptic Curve Diffie-Hellman Key Agreement with three different Key Derivation Function (KDF) routines. The inputs to the routines, and the routines themselves, were configured via properties on the [ECDiffieHellmanCng](#) object. But since not every routine read every input property, there was ample room for confusion on the part of the developer.

To address this in .NET Framework 4.6.2, the following three methods have been added to the [ECDiffieHellman](#) base class to more clearly represent these KDF routines and their inputs:

[+] Expand table

ECDiffieHellman method	Description
DeriveKeyFromHash(ECDiffieHellmanPublicKey, HashAlgorithmName, Byte[], Byte[])	Derives key material using the formula $\text{HASH}(\text{secretPrepend} \parallel x \parallel \text{secretAppend})$
DeriveKeyFromOrElse(ECDiffieHellmanPublicKey, HashAlgorithmName, Byte[], Byte[])	$\text{HASH}(\text{secretPrepend OrElse } x \text{ OrElse secretAppend})$ where x is the computed result of the EC Diffie-Hellman algorithm.
DeriveKeyFromHmac(ECDiffieHellmanPublicKey, HashAlgorithmName, Byte[], Byte[], Byte[])	Derives key material using the formula

ECDiffieHellman method	Description
	HMAC(hmacKey, secretPrepend x secretAppend)
	HMAC(hmacKey, secretPrepend OrElse x OrElse secretAppend)
	where x is the computed result of the EC Diffie-Hellman algorithm.
DeriveKeyTls(ECDiffieHellmanPublicKey, Byte[], Byte[])	Derives key material using the TLS pseudo-random function (PRF) derivation algorithm.

Support for persisted-key symmetric encryption

The Windows cryptography library (CNG) added support for storing persisted symmetric keys and using hardware-stored symmetric keys, and .NET Framework 4.6.2 made it possible for developers to make use of this feature. Since the notion of key names and key providers is implementation-specific, using this feature requires utilizing the constructor of the concrete implementation types instead of the preferred factory approach (such as calling `Aes.Create`).

Persisted-key symmetric encryption support exists for the AES ([AesCng](#)) and 3DES ([TripleDESCng](#)) algorithms. For example:

C#

```
public static byte[] EncryptDataWithPersistedKey(byte[] data, byte[] iv)
{
    using (Aes aes = new AesCng("AesDemoKey",
        CngProvider.MicrosoftSoftwareKeyStorageProvider))
    {
        aes.IV = iv;

        // Using the zero-argument overload is required to make use of the
        // persisted key
        using (ICryptoTransform encryptor = aes.CreateEncryptor())
        {
            if (!encryptor.CanTransformMultipleBlocks)
            {
                throw new InvalidOperationException("This is a sample, this
                    case wasn't handled...");
            }

            return encryptor.TransformFinalBlock(data, 0, data.Length);
        }
    }
}
```

```
    }  
}
```

SignedXml support for SHA-2 hashing

.NET Framework 4.6.2 adds support to the [SignedXml](#) class for RSA-SHA256, RSA-SHA384, and RSA-SHA512 PKCS#1 signature methods, and SHA256, SHA384, and SHA512 reference digest algorithms.

The URI constants are all exposed on [SignedXml](#):

[+] [Expand table](#)

SignedXml field	Constant
XmlDsigSHA256Url	" http://www.w3.org/2001/04/xmlenc#sha256 "
XmlDsigRSASHA256Url	" http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 "
XmlDsigSHA384Url	" http://www.w3.org/2001/04/xmldsig-more#sha384 "
XmlDsigRSASHA384Url	" http://www.w3.org/2001/04/xmldsig-more#rsa-sha384 "
XmlDsigSHA512Url	" http://www.w3.org/2001/04/xmlenc#sha512 "
XmlDsigRSASHA512Url	" http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 "

Any programs that have registered a custom [SignatureDescription](#) handler into [CryptoConfig](#) to add support for these algorithms will continue to function as they did in the past, but since there are now platform defaults, the [CryptoConfig](#) registration is no longer necessary.

SqlClient

.NET Framework Data Provider for SQL Server ([System.Data.SqlClient](#)) includes the following new features in .NET Framework 4.6.2:

Connection pooling and timeouts with Azure SQL databases

When connection pooling is enabled and a timeout or other login error occurs, an exception is cached, and the cached exception is thrown on any subsequent connection attempt for the next 5 seconds to 1 minute. For more information, see [SQL Server Connection Pooling \(ADO.NET\)](#).

This behavior is not desirable when connecting to Azure SQL Databases, since connection attempts can fail with transient errors that are typically recovered quickly. To

better optimize the connection retry experience, the connection pool blocking period behavior is removed when connections to Azure SQL Databases fail.

The addition of the new `PoolBlockingPeriod` keyword lets you select the blocking period best suited for your app. Values include:

[Auto](#)

The connection pool blocking period for an application that connects to an Azure SQL Database is disabled, and the connection pool blocking period for an application that connects to any other SQL Server instance is enabled. This is the default value. If the Server endpoint name ends with any of the following, they are considered Azure SQL Databases:

- `.database.windows.net`
- `.database.chinacloudapi.cn`
- `.database.usgovcloudapi.net`
- `.database.cloudapi.de`

[AlwaysBlock](#)

The connection pool blocking period is always enabled.

[NeverBlock](#)

The connection pool blocking period is always disabled.

Enhancements for Always Encrypted

SQLClient introduces two enhancements for Always Encrypted:

- To improve performance of parameterized queries against encrypted database columns, encryption metadata for query parameters is now cached. With the `SqlConnection.ColumnEncryptionQueryMetadataCacheEnabled` property set to `true` (which is the default value), if the same query is called multiple times, the client retrieves parameter metadata from the server only once.
- Column encryption key entries in the key cache are now evicted after a configurable time interval, set using the `SqlConnection.ColumnEncryptionKeyCacheTtl` property.

Windows Communication Foundation

In .NET Framework 4.6.2, Windows Communication Foundation has been enhanced in the following areas:

WCF transport security support for certificates stored using CNG

WCF transport security supports certificates stored using the Windows cryptography library (CNG). In .NET Framework 4.6.2, this support is limited to using certificates with a public key that has an exponent no more than 32 bits in length. When an application targets .NET Framework 4.6.2, this feature is on by default.

For applications that target .NET Framework 4.6.1 and earlier but are running on .NET Framework 4.6.2, this feature can be enabled by adding the following line to the `<runtime>` section of the app.config or web.config file.

XML

```
<AppContextSwitchOverrides  
    value="Switch.System.IdentityModel.DisableCngCertificates=false"  
/>
```

This can also be done programmatically with code like the following:

C#

```
private const string DisableCngCertificates =  
@$"Switch.System.IdentityModel.DisableCngCertificates";  
AppContext.SetSwitch(disableCngCertificates, false);
```

Better support for multiple daylight saving time adjustment rules by the DataContractJsonSerializer class

Customers can use an application configuration setting to determine whether the `DataContractJsonSerializer` class supports multiple adjustment rules for a single time zone. This is an opt-in feature. To enable it, add the following setting to your app.config file:

XML

```
<runtime>  
    <AppContextSwitchOverrides  
    value="Switch.System.Runtime.Serialization.DoNotUseTimeZoneInfo=false" />  
</runtime>
```

When this feature is enabled, a `DataContractJsonSerializer` object uses the `TimeZoneInfo` type instead of the `TimeZone` type to deserialize date and time data. `TimeZoneInfo`

supports multiple adjustment rules, which makes it possible to work with historic time zone data; [TimeZone](#) does not.

For more information on the [TimeZoneInfo](#) structure and time zone adjustments, see [Time Zone Overview](#).

NetNamedPipeBinding best match

WCF has a new app setting that can be set on client applications to ensure they always connect to the service listening on the URI that best matches the one that they request. With this app setting set to `false` (the default), it is possible for clients using [NetNamedPipeBinding](#) to attempt to connect to a service listening on a URI that is a substring of the requested URI.

For example, a client tries to connect to a service listening at `net.pipe://localhost/Service1`, but a different service on that machine running with administrator privilege is listening at `net.pipe://localhost`. With this app setting set to `false`, the client would attempt to connect to the wrong service. After setting the app setting to `true`, the client will always connect to the best matching service.

ⓘ Note

Clients using [NetNamedPipeBinding](#) find services based on the service's base address (if it exists) rather than the full endpoint address. To ensure this setting always works the service should use a unique base address.

To enable this change, add the following app setting to your client application's App.config or Web.config file:

XML

```
<configuration>
  <appSettings>
    <add key="wcf:useBestMatchNamedPipeUri" value="true" />
  </appSettings>
</configuration>
```

SSL 3.0 is not a default protocol

When using NetTcp with transport security and a credential type of certificate, SSL 3.0 is no longer a default protocol used for negotiating a secure connection. In most cases, there should be no impact to existing apps, because TLS 1.0 is included in the protocol list for NetTcp. All existing clients should be able to negotiate a connection using at least

TLS 1.0. If Ssl3 is required, use one of the following configuration mechanisms to add it to the list of negotiated protocols.

- The [SslStreamSecurityBindingElement.SslProtocols](#) property
- The [TcpTransportSecurity.SslProtocols](#) property
- The [`<transport>`](#) section of the [`<netTcpBinding>`](#) section
- The [`<sslStreamSecurity>`](#) section of the [`<customBinding>`](#) section

Windows Presentation Foundation (WPF)

In .NET Framework 4.6.2, Windows Presentation Foundation has been enhanced in the following areas:

Group sorting

An application that uses a [CollectionView](#) object to group data can now explicitly declare how to sort the groups. Explicit sorting addresses the problem of non-intuitive ordering that occurs when an app dynamically adds or removes groups, or when it changes the value of item properties involved in grouping. It can also improve the performance of the group creation process by moving comparisons of the grouping properties from the sort of the full collection to the sort of the groups.

To support group sorting, the new [GroupDescription.SortDescriptions](#) and [GroupDescription.CustomSort](#) properties describe how to sort the collection of groups produced by the [GroupDescription](#) object. This is analogous to the way the identically named [ListCollectionView](#) properties describe how to sort the data items.

Two new static properties of the [PropertyGroupDescription](#) class, [CompareNameAscending](#) and [CompareNameDescending](#), can be used for the most common cases.

For example, the following XAML groups data by age, sort the age groups in ascending order, and group the items within each age group by last name.

XAML

```
<GroupDescriptions>
    <PropertyGroupDescription
        PropertyName="Age"
        CustomSort=
            "{x:Static PropertyGroupDescription.CompareNamesAscending}" />
    </PropertyGroupDescription>
</GroupDescriptions>
```

```
<SortDescriptions>
  <SortDescription PropertyName="LastName"/>
</SortDescriptions>
```

Touch keyboard support

Touch keyboard support enables focus tracking in WPF applications by automatically invoking and dismissing the touch Keyboard in Windows 10 when the touch input is received by a control that can take textual input.

In previous versions of .NET Framework, WPF applications can't opt into the focus tracking without disabling WPF pen/touch gesture support. As a result, WPF applications must choose between full WPF touch support or rely on Windows mouse promotion.

Per-monitor DPI

To support the recent proliferation of high-DPI and hybrid-DPI environments for WPF apps, WPF in .NET Framework 4.6.2 enables per-monitor awareness. See the [samples and developer guide](#) ↗ on GitHub for more information about how to enable your WPF app to become per-monitor DPI aware.

In previous versions of .NET Framework, WPF apps are system-DPI aware. In other words, the application's UI is scaled by the OS as appropriate, depending on the DPI of the monitor on which the app is rendered.

For apps running under .NET Framework 4.6.2, you can disable per-monitor DPI changes in WPF apps by adding a configuration statement to the `<runtime>` section of your application configuration file, as follows:

XML

```
<runtime>
  <AppContextSwitchOverrides
    value="Switch.System.Windows.DoNotScaleForDpiChanges=false"/>
</runtime>
```

Windows Workflow Foundation (WF)

In .NET Framework 4.6.2, Windows Workflow Foundation has been enhanced in the following area:

[Support for C# expressions and IntelliSense in the Rehosted WF Designer](#)

Starting with .NET Framework 4.5, WF supports C# expressions in both the Visual Studio Designer and in code workflows. The Rehosted Workflow Designer is a key feature of WF that allows for the Workflow Designer to be in an application outside Visual Studio (for example, in WPF). Windows Workflow Foundation provides the ability to support C# expressions and IntelliSense in the Rehosted Workflow Designer. For more information, see the [Windows Workflow Foundation blog](#).

Availability of IntelliSense when a customer rebuilds a workflow project from Visual Studio In versions of the .NET Framework prior to 4.6.2, WF Designer IntelliSense is broken when a customer rebuilds a workflow project from Visual Studio. While the project build is successful, the workflow types are not found on the designer, and warnings from IntelliSense for the missing workflow types appear in the **Error List** window. .NET Framework 4.6.2 addresses this issue and makes IntelliSense available.

Workflow V1 applications with Workflow Tracking on now run under FIPS-mode

Machines with FIPS Compliance Mode enabled can now successfully run a workflow Version 1-style application with Workflow tracking on. To enable this scenario, you must make the following change to your app.config file:

XML

```
<add key="microsoft:WorkflowRuntime:FIPSRequired" value="true" />
```

If this scenario is not enabled, running the application continues to generate an exception with the message, "This implementation is not part of the Windows Platform FIPS validated cryptographic algorithms."

Workflow Improvements when using Dynamic Update with Visual Studio Workflow Designer

The Workflow Designer, FlowChart Activity Designer, and other Workflow Activity Designers now successfully load and display workflows that have been saved after calling the [DynamicUpdateServices.PrepareForUpdate](#) method. In versions of the .NET Framework before .NET Framework 4.6.2, loading a XAML file in Visual Studio for a workflow that has been saved after calling [DynamicUpdateServices.PrepareForUpdate](#) can result in the following issues:

- The Workflow Designer can't load the XAML file correctly (when the [ViewStateData.Id](#) is at the end of the line).
- Flowchart Activity Designer or other Workflow Activity Designers may display all objects in their default locations as opposed to attached property values.

ClickOnce

ClickOnce has been updated to support TLS 1.1 and TLS 1.2 in addition to the 1.0 protocol, which it already supports. ClickOnce automatically detects which protocol is required; no extra steps within the ClickOnce application are required to enable TLS 1.1 and 1.2 support.

Converting Windows Forms and WPF apps to UWP apps

Windows now offers capabilities to bring existing Windows desktop apps, including WPF and Windows Forms apps, to the Universal Windows Platform (UWP). This technology acts as a bridge by enabling you to gradually migrate your existing code base to UWP, thereby bringing your app to all Windows 10 devices.

Converted desktop apps gain an app identity similar to the app identity of UWP apps, which makes UWP APIs accessible to enable features such as Live Tiles and notifications. The app continues to behave as before and runs as a full trust app. Once the app is converted, an app container process can be added to the existing full trust process to add an adaptive user interface. When all functionality is moved to the app container process, the full trust process can be removed and the new UWP app can be made available to all Windows 10 devices.

Debugging improvements

The *unmanaged debugging API* has been enhanced in .NET Framework 4.6.2 to perform additional analysis when a [NullReferenceException](#) is thrown so that it is possible to determine which variable in a single line of source code is `null`. To support this scenario, the following APIs have been added to the unmanaged debugging API.

- The [ICorDebugCode4](#), [ICorDebugVariableHome](#), and [ICorDebugVariableHomeEnum](#) interfaces, which expose the native homes of managed variables. This enables debuggers to do some code flow analysis when a [NullReferenceException](#) occurs and to work backwards to determine the managed variable that corresponds to the native location that was `null`.
- The [ICorDebugType2::GetTypeID](#) method provides a mapping for [ICorDebugType](#) to [COR_TYPEID](#), which allows the debugger to obtain a [COR_TYPEID](#) without an instance of the [ICorDebugType](#). Existing APIs on [COR_TYPEID](#) can then be used to determine the class layout of the type.

What's new in .NET Framework 4.6.1

.NET Framework 4.6.1 includes new features in the following areas:

- [Cryptography](#)
- [ADO.NET](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Workflow Foundation](#)
- [Profiling](#)
- [NGen](#)

For more information on .NET Framework 4.6.1, see the following topics:

- [.NET Framework 4.6.1 list of changes ↗](#)
- [Application Compatibility in 4.6.1](#)
- [.NET Framework API diff ↗](#) (on GitHub)

Cryptography: Support for X509 certificates containing ECDSA

.NET Framework 4.6 added RSACng support for X509 certificates. .NET Framework 4.6.1 adds support for ECDSA (Elliptic Curve Digital Signature Algorithm) X509 certificates.

ECDSA offers better performance and is a more secure cryptography algorithm than RSA, providing an excellent choice where Transport Layer Security (TLS) performance and scalability is a concern. The .NET Framework implementation wraps calls into existing Windows functionality.

The following example code shows how easy it is to generate a signature for a byte stream by using the new support for ECDSA X509 certificates included in .NET Framework 4.6.1.

C#

```
using System;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;

public class Net461Code
{
    public static byte[] SignECDsaSha512(byte[] data, X509Certificate2 cert)
    {
        using (ECDsa privateKey = cert.GetECDsaPrivateKey())
```

```

        {
            return privateKey.SignData(data, HashAlgorithmName.SHA512);
        }
    }

    public static byte[] SignECDsaSha512(byte[] data, ECDsa privateKey)
    {
        return privateKey.SignData(data, HashAlgorithmName.SHA512);
    }
}

```

This offers a marked contrast to the code needed to generate a signature in .NET Framework 4.6.

C#

```

using System;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;

public class Net46Code
{
    public static byte[] SignECDsaSha512(byte[] data, X509Certificate2 cert)
    {
        // This would require using cert.Handle and a series of p/invokes to
        // get at the
        // underlying key, then passing that to a CngKey object, and passing
        // that to
        // new ECDsa(CngKey). It's a lot of work.
        throw new Exception("That's a lot of work...");
    }

    public static byte[] SignECDsaSha512(byte[] data, ECDsa privateKey)
    {
        // This way works, but SignData probably better matches what you
        // want.
        using (SHA512 hasher = SHA512.Create())
        {
            byte[] signature1 =
privateKey.SignHash(hasher.ComputeHash(data));
        }

        // This might not be the ECDsa you got!
        ECDsaCng ecDsaCng = (ECDsaCng)privateKey;
        ecDsaCng.HashAlgorithm = CngAlgorithm.Sha512;
        return ecDsaCng.SignData(data);
    }
}

```

The following have been added to ADO.NET:

Always Encrypted support for hardware protected keys

ADO.NET now supports storing Always Encrypted column master keys natively in Hardware Security Modules (HSMs). With this support, customers can leverage asymmetric keys stored in HSMs without having to write custom column master key store providers and registering them in applications.

Customers need to install the HSM vendor-provided CSP provider or CNG key store providers on the app servers or client computers in order to access Always Encrypted data protected with column master keys stored in an HSM.

Improved [MultiSubnetFailover](#) connection behavior for AlwaysOn

SqlClient now automatically provides faster connections to an AlwaysOn Availability Group (AG). It transparently detects whether your application is connecting to an AlwaysOn availability group (AG) on a different subnet and quickly discovers the current active server and provides a connection to the server. Prior to this release, an application had to set the connection string to include `"MultisubnetFailover=true"` to indicate that it was connecting to an AlwaysOn Availability Group. Without setting the connection keyword to `true`, an application might experience a timeout while connecting to an AlwaysOn Availability Group. With this release, an application does *not* need to set [MultiSubnetFailover](#) to `true` anymore. For more information about SqlClient support for Always On Availability Groups, see [SqlClient Support for High Availability, Disaster Recovery](#).

Windows Presentation Foundation (WPF)

Windows Presentation Foundation includes a number of improvements and changes.

Improved performance

The delay in firing touch events has been fixed in .NET Framework 4.6.1. In addition, typing in a [RichTextBox](#) control no longer ties up the render thread during fast input.

Spell checking improvements

The spell checker in WPF has been updated on Windows 8.1 and later versions to leverage operating system support for spell-checking additional languages. There is no change in functionality on Windows versions prior to Windows 8.1.

As in previous versions of .NET Framework, the language for a [TextBox](#) control or a [RichTextBox](#) block is detected by looking for information in the following order:

- `xml:lang`, if it is present.
- Current input language.
- Current culture.

For more information on language support in WPF, see the [WPF blog post on .NET Framework 4.6.1 features](#).

Additional support for per-user custom dictionaries

In .NET Framework 4.6.1, WPF recognizes custom dictionaries that are registered globally. This capability is available in addition to the ability to register them per-control.

In previous versions of WPF, custom dictionaries did not recognize Excluded Words and AutoCorrect lists. They are supported on Windows 8.1 and Windows 10 through the use of files that can be placed under the `%AppData%\Microsoft\Spelling\<language tag>` directory. The following rules apply to these files:

- The files should have extensions of .dic (for added words), .exc (for excluded words), or .acl (for AutoCorrect).
- The files should be UTF-16 LE plaintext that starts with the Byte Order Mark (BOM).
- Each line should consist of a word (in the added and excluded word lists), or an autocorrect pair with the words separated by a vertical bar ("|") (in the AutoCorrect word list).
- These files are considered read-only and are not modified by the system.

Note

These new file-formats are not directly supported by the WPF spell checking APIs, and the custom dictionaries supplied to WPF in applications should continue to use .lex files.

Samples

There are a number of WPF samples on the [Microsoft/WPF-Samples](#) GitHub repository. Help us improve our samples by sending us a pull-request or opening a [GitHub issue](#).

DirectX extensions

WPF includes a [NuGet package](#) that provides new implementations of [D3DImage](#) that make it easy for you to interoperate with DX10 and Dx11 content. The code for this package has been open sourced and is available [on GitHub](#).

Windows Workflow Foundation: Transactions

The [Transaction.EnlistPromotableSinglePhase](#) method can now use a distributed transaction manager other than MSDTC to promote the transaction. You do this by specifying a GUID transaction promoter identifier to the new [Transaction.EnlistPromotableSinglePhase\(IPromotableSinglePhaseNotification, Guid\)](#) overload . If this operation is successful, there are limitations placed on the capabilities of the transaction. Once a non-MSDTC transaction promoter is enlisted, the following methods throw a [TransactionPromotionException](#) because these methods require promotion to MSDTC:

- [Transaction.EnlistDurable](#)
- [TransactionInterop.GetDtcTransaction](#)
- [TransactionInterop.GetExportCookie](#)
- [TransactionInterop.GetTransmitterPropagationToken](#)

Once a non-MSDTC transaction promoter is enlisted, it must be used for future durable enlistments by using protocols that it defines. The [Guid](#) of the transaction promoter can be obtained by using the [PromoterType](#) property. When the transaction promotes, the transaction promoter provides a [Byte](#) array that represents the promoted token. An application can obtain the promoted token for a non-MSDTC promoted transaction with the [GetPromotedToken](#) method.

Users of the new

[Transaction.EnlistPromotableSinglePhase\(IPromotableSinglePhaseNotification, Guid\)](#) overload must follow a specific call sequence in order for the promotion operation to complete successfully. These rules are documented in the method's documentation.

Profiling

The unmanaged profiling API has been enhanced as follows:

- Better support for accessing PDBs in the [ICorProfilerInfo7](#) interface.

In ASP.NET Core, it is becoming much more common for assemblies to be compiled in-memory by Roslyn. For developers making profiling tools, this means

that PDBs that historically were serialized on disk may no longer be present.

Profiler tools often use PDBs to map code back to source lines for tasks such as code coverage or line-by-line performance analysis. The [ICorProfilerInfo7](#) interface now includes two new methods, [ICorProfilerInfo7::GetInMemorySymbolsLength](#) and [ICorProfilerInfo7::ReadInMemorySymbols](#), to provide these profiler tools with access to the in-memory PDB data. By using the new APIs, a profiler can obtain the contents of an in-memory PDB as a byte array and then process it or serialize it to disk.

- Better instrumentation with the ICorProfiler interface.

Profilers that are using the [ICorProfiler](#) APIs ReJit functionality for dynamic instrumentation can now modify some metadata. Previously such tools could instrument IL at any time, but metadata could only be modified at module load time. Because IL refers to metadata, this limited the kinds of instrumentation that could be done. We have lifted some of those limits by adding the [ICorProfilerInfo7::ApplyMetaData](#) method to support a subset of metadata edits after the module loads, in particular by adding new [AssemblyRef](#), [TypeRef](#), [TypeSpec](#), [MemberRef](#), [MemberSpec](#), and [UserString](#) records. This change makes a much broader range of on-the-fly instrumentation possible.

Native Image Generator (NGEN) PDBs

Cross-machine event tracing allows customers to profile a program on Machine A and look at the profiling data with source line mapping on Machine B. Using previous versions of .NET Framework, the user would copy all the modules and native images from the profiled machine to the analysis machine that contains the IL PDB to create the source-to-native mapping. While this process may work well when the files are relatively small, such as for phone applications, the files can be very large on desktop systems and require significant time to copy.

With Ngen PDBs, NGen can create a PDB that contains the IL-to-native mapping without a dependency on the IL PDB. In our cross-machine event tracing scenario, all that is needed is to copy the native image PDB that is generated by Machine A to Machine B and to use [Debug Interface Access APIs](#) to read the IL PDB's source-to-IL mapping and the native image PDB's IL-to-native mapping. Combining both mappings provides a source-to-native mapping. Since the native image PDB is much smaller than all the modules and native images, the process of copying from Machine A to Machine B is much faster.

What's new in .NET 2015

.NET 2015 introduces .NET Framework 4.6 and .NET Core. Some new features apply to both, and other features are specific to .NET Framework 4.6 or .NET Core.

- **ASP.NET Core**

.NET 2015 includes ASP.NET Core, which is a lean .NET implementation for building modern cloud-based apps. ASP.NET Core is modular so you can include only those features that are needed in your application. It can be hosted on IIS or self-hosted in a custom process, and you can run apps with different versions of the .NET Framework on the same server. It includes a new environment configuration system that is designed for cloud deployment.

MVC, Web API, and Web Pages are unified into a single framework called MVC 6. You build ASP.NET Core apps through tools in Visual Studio 2015 or later. Your existing applications will work on the new .NET Framework; however to build an app that uses MVC 6 or SignalR 3, you must use the project system in Visual Studio 2015 or later.

For information, see [ASP.NET Core](#).

- **ASP.NET Updates**

- **Task-based API for Asynchronous Response Flushing**

ASP.NET now provides a simple task-based API for asynchronous response flushing, [HttpResponse.FlushAsync](#), that allows responses to be flushed asynchronously by using your language's `async/await` support.

- **Model binding supports task-returning methods**

In .NET Framework 4.5, ASP.NET added the Model Binding feature that enabled an extensible, code-focused approach to CRUD-based data operations in Web Forms pages and user controls. The Model Binding system now supports [Task](#)-returning model binding methods. This feature allows Web Forms developers to get the scalability benefits of `async` with the ease of the data-binding system when using newer versions of ORMs, including the Entity Framework.

Async model binding is controlled by the `aspnet:EnableAsyncModelBinding` configuration setting.

XML

```
<appSettings>
    <add key=" aspnet:EnableAsyncModelBinding" value="true|false" />
</appSettings>
```

On apps the target .NET Framework 4.6, it defaults to `true`. On apps running on .NET Framework 4.6 that target an earlier version of .NET Framework, it is `false` by default. It can be enabled by setting the configuration setting to `true`.

- **HTTP/2 Support (Windows 10)**

[HTTP/2](#) is a new version of the HTTP protocol that provides much better connection utilization (fewer round-trips between client and server), resulting in lower latency web page loading for users. Web pages (as opposed to services) benefit the most from HTTP/2, since the protocol optimizes for multiple artifacts being requested as part of a single experience. HTTP/2 support has been added to ASP.NET in .NET Framework 4.6. Because networking functionality exists at multiple layers, new features were required in Windows, in IIS, and in ASP.NET to enable HTTP/2. You must be running on Windows 10 to use HTTP/2 with ASP.NET.

HTTP/2 is also supported and on by default for Windows 10 Universal Windows Platform (UWP) apps that use the [System.Net.Http.HttpClient](#) API.

In order to provide a way to use the [PUSH_PROMISE](#) feature in ASP.NET applications, a new method with two overloads, [PushPromise\(String\)](#) and [PushPromise\(String, String, NameValueCollection\)](#), has been added to the [HttpResponse](#) class.

① Note

While ASP.NET Core supports HTTP/2, support for the PUSH PROMISE feature has not yet been added.

The browser and the web server (IIS on Windows) do all the work. You don't have to do any heavy-lifting for your users.

Most of the [major browsers support HTTP/2](#), so it's likely that your users will benefit from HTTP/2 support if your server supports it.

- **Support for the Token Binding Protocol**

Microsoft and Google have been collaborating on a new approach to authentication, called the [Token Binding Protocol](#). The premise is that authentication tokens (in your browser cache) can be stolen and used by criminals to access otherwise secure resources (for example, your bank account) without requiring your password or any other privileged knowledge. The new protocol aims to mitigate this problem.

The Token Binding Protocol will be implemented in Windows 10 as a browser feature. ASP.NET apps will participate in the protocol, so that authentication tokens are validated to be legitimate. The client and the server implementations establish the end-to-end protection specified by the protocol.

- **Randomized string hash algorithms**

.NET Framework 4.5 introduced a [randomized string hash algorithm](#). However, it was not supported by ASP.NET because of some ASP.NET features depended on a stable hash code. In .NET Framework 4.6, randomized string hash algorithms are now supported. To enable this feature, use the `aspnet:UseRandomizedStringHashAlgorithm` config setting.

XML

```
<appSettings>
    <add key="aspnet:UseRandomizedStringHashAlgorithm"
        value="true|false" />
</appSettings>
```

- **ADO.NET**

ADO .NET now supports the Always Encrypted feature available in SQL Server 2016. With Always Encrypted, SQL Server can perform operations on encrypted data, and best of all the encryption key resides with the application inside the customer's trusted environment and not on the server. Always Encrypted secures customer data so DBAs do not have access to plain text data. Encryption and decryption of data happens transparently at the driver level, minimizing changes that have to be made to existing applications. For details, see [Always Encrypted \(Database Engine\)](#) and [Always Encrypted \(client development\)](#).

- **64-bit JIT Compiler for managed code**

.NET Framework 4.6 features a new version of the 64-bit JIT compiler (originally code-named RyuJIT). The new 64-bit compiler provides significant performance improvements over the older 64-bit JIT compiler. The new 64-bit compiler is enabled for 64-bit processes running on top of .NET Framework 4.6. Your app will run in a 64-bit process if it is compiled as 64-bit or AnyCPU and is running on a 64-bit operating system. While care has been taken to make the transition to the new compiler as transparent as possible, changes in behavior are possible.

The new 64-bit JIT compiler also includes hardware SIMD acceleration features when coupled with SIMD-enabled types in the [System.Numerics](#) namespace, which can yield good performance improvements.

- **Assembly loader improvements**

The assembly loader now uses memory more efficiently by unloading IL assemblies after a corresponding NGEN image is loaded. This change decreases virtual memory, which is particularly beneficial for large 32-bit apps (such as Visual Studio), and also saves physical memory.

- **Base class library changes**

Many new APIs have been added around to .NET Framework 4.6 to enable key scenarios. These include the following changes and additions:

- **IReadOnlyCollection<T> implementations**

Additional collections implement [IReadOnlyCollection<T>](#) such as [Queue<T>](#) and [Stack<T>](#).

- **CultureInfo.CurrentCulture and CultureInfo.CurrentUICulture**

The [CultureInfo.CurrentCulture](#) and [CultureInfo.CurrentUICulture](#) properties are now read-write rather than read-only. If you assign a new [CultureInfo](#) object to these properties, the current thread culture defined by the

`Thread.CurrentThread.CurrentCulture` property and the current UI thread culture defined by the `Thread.CurrentThread.CurrentUICulture` properties also change.

- **Enhancements to garbage collection (GC)**

The [GC](#) class now includes [TryStartNoGCRegion](#) and [EndNoGCRegion](#) methods that allow you to disallow garbage collection during the execution of a critical path.

A new overload of the [GC.Collect\(Int32, GCCollectionMode, Boolean, Boolean\)](#) method allows you to control whether both the small object heap and the large object heap are swept and compacted or swept only.

- **SIMD-enabled types**

The [System.Numerics](#) namespace now includes a number of SIMD-enabled types, such as [Matrix3x2](#), [Matrix4x4](#), [Plane](#), [Quaternion](#), [Vector2](#), [Vector3](#), and [Vector4](#).

Because the new 64-bit JIT compiler also includes hardware SIMD acceleration features, there are especially significant performance improvements when using the SIMD-enabled types with the new 64-bit JIT compiler.

- Cryptography updates

The [System.Security.Cryptography](#) API is being updated to support the [Windows CNG cryptography APIs](#). Previous versions of the .NET Framework have relied entirely on an [earlier version of the Windows Cryptography APIs](#) as the basis for the [System.Security.Cryptography](#) implementation. We have had requests to support the CNG API, since it supports [modern cryptography algorithms](#), which are important for certain categories of apps.

.NET Framework 4.6 includes the following new enhancements to support the Windows CNG cryptography APIs:

- A set of extension methods for X509 Certificates,

```
System.Security.Cryptography.X509Certificates.RSACertificateExtensions.GetRSAPublicKey(System.Security.Cryptography.X509Certificates.X509Certificate2) and
```

```
System.Security.Cryptography.X509Certificates.RSACertificateExtensions.GetRSAPrivateKey(System.Security.Cryptography.X509Certificates.X509Certificate2), that return a CNG-based implementation rather than a CAPI-based implementation when possible. (Some smartcards, etc., still require CAPI, and the APIs handle the fallback).
```

- The [System.Security.Cryptography.RSACng](#) class, which provides a CNG implementation of the RSA algorithm.
- Enhancements to the RSA API so that common actions no longer require casting. For example, encrypting data using an [X509Certificate2](#) object requires code like the following in previous versions of .NET Framework.

```
C#
```

```
RSACryptoServiceProvider rsa =
(RSACryptoServiceProvider)cert.PrivateKey;
byte[] oaepEncrypted = rsa.Encrypt(data, true);
byte[] pkcs1Encrypted = rsa.Encrypt(data, false);
```

Code that uses the new cryptography APIs in .NET Framework 4.6 can be rewritten as follows to avoid the cast.

```
C#
```

```
RSA rsa = cert.GetRSAPrivateKey();
if (rsa == null)
    throw new InvalidOperationException("An RSA certificate was
expected");
```

```
byte[] oaepEncrypted = rsa.Encrypt(data,
RSAEncryptionPadding.OaepSHA1);
byte[] pkcs1Encrypted = rsa.Encrypt(data,
RSAEncryptionPadding.Pkcs1);
```

- **Support for converting dates and times to or from Unix time**

The following new methods have been added to the [DateTimeOffset](#) structure to support converting date and time values to or from Unix time:

- [DateTimeOffset.FromUnixTimeSeconds](#)
 - [DateTimeOffset.FromUnixTimeMilliseconds](#)
 - [DateTimeOffset.ToUnixTimeSeconds](#)
 - [DateTimeOffset.ToUnixTimeMilliseconds](#)
- **Compatibility switches**

The [AppContext](#) class adds a new compatibility feature that enables library writers to provide a uniform opt-out mechanism for new functionality for their users. It establishes a loosely coupled contract between components in order to communicate an opt-out request. This capability is typically important when a change is made to existing functionality. Conversely, there is already an implicit opt-in for new functionality.

With [AppContext](#), libraries define and expose compatibility switches, while code that depends on them can set those switches to affect the library behavior. By default, libraries provide the new functionality, and they only alter it (that is, they provide the previous functionality) if the switch is set.

An application (or a library) can declare the value of a switch (which is always a [Boolean](#) value) that a dependent library defines. The switch is always implicitly `false`. Setting the switch to `true` enables it. Explicitly setting the switch to `false` provides the new behavior.

C#

```
AppContext.SetSwitch("Switch.AmazingLib.ThrowOnException", true);
```

The library must check if a consumer has declared the value of the switch and then appropriately act on it.

C#

```
if (!ApplicationContext.TryGetSwitch("Switch.AmazingLib.ThrowOnException",
    out shouldThrow))
{
    // This is the case where the switch value was not set by the
    // application.
    // The library can choose to get the value of shouldThrow by
    // other means.
    // If no overrides nor default values are specified, the value
    // should be 'false'.
    // A false value implies the latest behavior.
}

// The library can use the value of shouldThrow to throw exceptions
// or not.
if (shouldThrow)
{
    // old code
}
else
{
    // new code
}
```

It's beneficial to use a consistent format for switches, since they are a formal contract exposed by a library. The following are two obvious formats.

- *Switch.namespace.switchname*
- *Switch.library.switchname*
- **Changes to the task-based asynchronous pattern (TAP)**

For apps that target .NET Framework 4.6, [Task](#) and [Task<TResult>](#) objects inherit the culture and UI culture of the calling thread. The behavior of apps that target previous versions of .NET Framework, or that do not target a specific version of .NET Framework, is unaffected. For more information, see the "Culture and task-based asynchronous operations" section of the [CultureInfo](#) class topic.

The [System.Threading.AsyncLocal<T>](#) class allows you to represent ambient data that is local to a given asynchronous control flow, such as an `async` method. It can be used to persist data across threads. You can also define a callback method that is notified whenever the ambient data changes either because the [AsyncLocal<T>.Value](#) property was explicitly changed, or because the thread encountered a context transition.

Three convenience methods, [Task.CompletedTask](#), [Task.FromCanceled](#), and [Task.FromException](#), have been added to the task-based asynchronous pattern (TAP) to return completed tasks in a particular state.

The [NamedPipeClientStream](#) class now supports asynchronous communication with its new [ConnectAsync](#) method.

- **EventSource now supports writing to the Event log**

You now can use the [EventSource](#) class to log administrative or operational messages to the event log, in addition to any existing ETW sessions created on the machine. In the past, you had to use the [Microsoft.Diagnostics.Tracing.EventSource](#) NuGet package for this functionality. This functionality is now built-into .NET Framework 4.6.

Both the NuGet package and .NET Framework 4.6 have been updated with the following features:

- **Dynamic events**

Allows events defined "on the fly" without creating event methods.

- **Rich payloads**

Allows specially attributed classes and arrays as well as primitive types to be passed as a payload

- **Activity tracking**

Causes Start and Stop events to tag events between them with an ID that represents all currently active activities.

To support these features, the overloaded [Write](#) method has been added to the [EventSource](#) class.

- **Windows Presentation Foundation (WPF)**

- **HDPI improvements**

HDPI support in WPF is now better in .NET Framework 4.6. Changes have been made to layout rounding to reduce instances of clipping in controls with borders. By default, this feature is enabled only if your

[TargetFrameworkAttribute](#) is set to .NET Framework 4.6. Applications that target earlier versions of the framework but are running on .NET Framework 4.6 can opt in to the new behavior by adding the following line to the [`<runtime>`](#) section of the app.config file:

XML

```
<AppContextSwitchOverrides  
value="Switch.MS.Internal.DoNotApplyLayoutRoundingToMarginsAndBorder  
Thickness=false"  
/>
```

WPF windows straddling multiple monitors with different DPI settings (Multi-DPI setup) are now completely rendered without blacked-out regions. You can opt out of this behavior by adding the following line to the `<appSettings>` section of the app.config file to disable this new behavior:

XML

```
<add key="EnableMultiMonitorDisplayClipping" value="true"/>
```

Support for automatically loading the right cursor based on DPI setting has been added to [System.Windows.Input.Cursor](#).

- **Touch is better**

Customer reports on [Connect](#) that touch produces unpredictable behavior have been addressed in .NET Framework 4.6. The double tap threshold for Windows Store applications and WPF applications is now the same in Windows 8.1 and above.

- **Transparent child window support**

WPF in .NET Framework 4.6 supports transparent child windows in Windows 8.1 and above. This allows you to create non-rectangular and transparent child windows in your top-level windows. You can enable this feature by setting the [HwndSourceParameters.UsesPerPixelTransparency](#) property to `true`.

- **Windows Communication Foundation (WCF)**

- **SSL support**

WCF now supports SSL version TLS 1.1 and TLS 1.2, in addition to SSL 3.0 and TLS 1.0, when using NetTcp with transport security and client authentication. It is now possible to select which protocol to use, or to disable old lesser secure protocols. This can be done either by setting the [SslProtocols](#) property or by adding the following to a configuration file.

XML

```

<netTcpBinding>
    <binding>
        <security mode=
"None|Transport|Message|TransportWithMessageCredential" >
            <transport clientCredentialType="None|Windows|Certificate"
protectionLevel="None|Sign|EncryptAndSign"
sslProtocols="Ssl3|Tls1|Tls11|Tls12">
                </transport>
            </security>
        </binding>
    </netTcpBinding>

```

- **Sending messages using different HTTP connections**

WCF now allows users to ensure certain messages are sent using different underlying HTTP connections. There are two ways to do this:

- **Using a connection group name prefix**

Users can specify a string that WCF will use as a prefix for the connection group name. Two messages with different prefixes are sent using different underlying HTTP connections. You set the prefix by adding a key/value pair to the message's [Message.Properties](#) property. The key is "HttpTransportConnectionGroupNamePrefix"; the value is the desired prefix.

- **Using different channel factories**

Users can also enable a feature that ensures that messages sent using channels created by different channel factories will use different underlying HTTP connections. To enable this feature, users must set the following

`appSetting to true:`

XML

```

<appSettings>
    <add
key="wcf:httpTransportBinding:useUniqueConnectionPoolPerFactory"
value="true" />
</appSettings>

```

- **Windows Workflow Foundation (WWF)**

You can now specify the number of seconds a workflow service will hold on to an out-of-order operation request when there is an outstanding "non-protocol" bookmark before timing out the request. A "non-protocol" bookmark is a bookmark that is not related to outstanding Receive activities. Some activities

create non-protocol bookmarks within their implementation, so it may not be obvious that a non-protocol bookmark exists. These include State and Pick. So if you have a workflow service implemented with a state machine or containing a Pick activity, you will most likely have non-protocol bookmarks. You specify the interval by adding a line like the following to the `appSettings` section of your `app.config` file:

XML

```
<add key="microsoft:WorkflowServices:FilterResumeTimeoutInSeconds"
     value="60"/>
```

The default value is 60 seconds. If `value` is set to 0, out-of-order requests are immediately rejected with a fault with text that looks like this:

Console

```
Operation 'Request3|{http://tempuri.org/}IService' on service instance
with identifier '2b0667b6-09c8-4093-9d02-f6c67d534292' cannot be
performed at this time. Please ensure that the operations are performed
in the correct order and that the binding in use provides ordered
delivery guarantees.
```

This is the same message that you receive if an out-of-order operation message is received and there are no non-protocol bookmarks.

If the value of the `FilterResumeTimeoutInSeconds` element is non-zero, there are non-protocol bookmarks, and the timeout interval expires, the operation fails with a timeout message.

- **Transactions**

You can now include the distributed transaction identifier for the transaction that has caused an exception derived from `TransactionException` to be thrown. You do this by adding the following key to the `appSettings` section of your `app.config` file:

XML

```
<add
  key="Transactions:IncludeDistributedTransactionIdInExceptionMessage"
  value="true"/>
```

The default value is `false`.

- **Networking**

- **Socket reuse**

Windows 10 includes a new high-scalability networking algorithm that makes better use of machine resources by reusing local ports for outbound TCP connections. .NET Framework 4.6 supports the new algorithm, enabling .NET apps to take advantage of the new behavior. In previous versions of Windows, there was an artificial concurrent connection limit (typically 16,384, the default size of the dynamic port range), which could limit the scalability of a service by causing port exhaustion when under load.

In .NET Framework 4.6, two APIs have been added to enable port reuse, which effectively removes the 64 KB limit on concurrent connections:

- The [System.Net.Sockets.SocketOptionName](#) enumeration value.
- The [ServicePointManager.ReusePort](#) property.

By default, the [ServicePointManager.ReusePort](#) property is `false` unless the `HWRPortReuseOnSocketBind` value of the `HKLM\SOFTWARE\Microsoft\.NETFramework\v4.0.30319` registry key is set to 0x1. To enable local port reuse on HTTP connections, set the [ServicePointManager.ReusePort](#) property to `true`. This causes all outgoing TCP socket connections from [HttpClient](#) and [HttpWebRequest](#) to use a new Windows 10 socket option, [SO_REUSE_UNICASTPORT](#), that enables local port reuse.

Developers writing a sockets-only application can specify the [System.Net.Sockets.SocketOptionName](#) option when calling a method such as [Socket.SetSocketOption](#) so that outbound sockets reuse local ports during binding.

- **Support for international domain names and PunyCode**

A new property, [IdnHost](#), has been added to the [Uri](#) class to better support international domain names and PunyCode.

- **Resizing in Windows Forms controls.**

This feature has been expanded in .NET Framework 4.6 to include the [DomainUpDown](#), [NumericUpDown](#), [DataGridViewComboBoxColumn](#), [DataGridViewColumn](#) and [ToolStripSplitButton](#) types and the rectangle specified by the [Bounds](#) property used when drawing a [UITypeEditor](#).

This is an opt-in feature. To enable it, set the

`EnableWindowsFormsHighDpiAutoResizing` element to `true` in the application configuration (app.config) file:

XML

```
<appSettings>
  <add key="EnableWindowsFormsHighDpiAutoResizing" value="true" />
</appSettings>
```

- **Support for code page encodings**

.NET Core primarily supports the Unicode encodings and by default provides limited support for code page encodings. You can add support for code page encodings available in .NET Framework but unsupported in .NET Core by registering code page encodings with the [Encoding.RegisterProvider](#) method. For more information, see [System.Text.CodePagesEncodingProvider](#).

- **.NET Native**

Universal Windows Platform (UWP) apps that are written in C# or Visual Basic can take advantage of a new technology that compiles apps to native code rather than IL. This technology produces apps that have faster startup and execution times. For more information, see [Compiling Apps with .NET Native](#). For an overview of .NET Native that examines how it differs from both JIT compilation and NGEN and what that means for your code, see [.NET Native and Compilation](#).

Your apps are compiled to native code by default when you compile them with Visual Studio 2015 or later. For more information, see [Getting Started with .NET Native](#).

To support debugging .NET Native apps, a number of new interfaces and enumerations have been added to the unmanaged debugging API. For more information, see the [Debugging \(Unmanaged API Reference\)](#) topic.

- **Open-source .NET Framework packages**

.NET Core packages such as the immutable collections, [SIMD APIs](#), and networking APIs such as those found in the [System.Net.Http](#) namespace are now available as open-source packages on [GitHub](#). To access the code, see [.NET on GitHub](#). For more information and how to contribute to these packages, see [Introduction to .NET](#), [.NET Home Page on GitHub](#).

What's new in .NET Framework 4.5.2

- **New APIs for ASP.NET apps.** The new `HttpResponse.AddOnSendingHeaders` and `HttpResponseBase.AddOnSendingHeaders` methods let you inspect and modify response headers and status code as the response is being flushed to the client app. Consider using these methods instead of the `PreSendRequestHeaders` and `PreSendRequestContent` events; they are more efficient and reliable.

The `HostingEnvironment.QueueBackgroundWorkItem` method lets you schedule small background work items. ASP.NET tracks these items and prevents IIS from abruptly terminating the worker process until all background work items have completed. This method can't be called outside an ASP.NET managed app domain.

The new `HttpResponse.HeadersWritten` and `HttpResponseBase.HeadersWritten` properties return Boolean values that indicate whether the response headers have been written. You can use these properties to make sure that calls to APIs such as `HttpResponse.StatusCode` (which throw exceptions if the headers have been written) will succeed.

- **Resizing in Windows Forms controls.** This feature has been expanded. You can now use the system DPI setting to resize components of the following additional controls (for example, the drop-down arrow in combo boxes):
 - `ComboBox`
 - `ToolStripComboBox`
 - `ToolStripMenuItem`
 - `Cursor`
 - `DataGridView`
 - `DataGridViewComboBoxColumn`

This is an opt-in feature. To enable it, set the

`EnableWindowsFormsHighDpiAutoResizing` element to `true` in the application configuration (`app.config`) file:

XML

```
<appSettings>
    <add key="EnableWindowsFormsHighDpiAutoResizing" value="true" />
</appSettings>
```

- **New workflow feature.** A resource manager that's using the `EnlistPromutableSinglePhase` method (and therefore implementing the `IPromutableSinglePhaseNotification` interface) can use the new `Transaction.PromoteAndEnlistDurable` method to request the following:

- Promote the transaction to a Microsoft Distributed Transaction Coordinator (MSDTC) transaction.
- Replace [IPromotableSinglePhaseNotification](#) with an [ISinglePhaseNotification](#), which is a durable enlistment that supports single phase commits.

This can be done within the same app domain, and doesn't require any extra unmanaged code to interact with MSDTC to perform the promotion. The new method can be called only when there's an outstanding call from [System.Transactions](#) to the [IPromotableSinglePhaseNotification](#) `Promote` method that's implemented by the promotable enlistment.

- **Profiling improvements.** The following new unmanaged profiling APIs provide more robust profiling:
 - [COR_PRF_ASSEMBLY_REFERENCE_INFO Structure](#)
 - [COR_PRF_HIGH_MONITOR Enumeration](#)
 - [GetAssemblyReferences Method](#)
 - [GetEventMask2 Method](#)
 - [SetEventMask2 Method](#)
 - [AddAssemblyReference Method](#)

Previous [ICorProfiler](#) implementations supported lazy loading of dependent assemblies. The new profiling APIs require dependent assemblies that are injected by the profiler to be loadable immediately, instead of being loaded after the app is fully initialized. This change doesn't affect users of the existing [ICorProfiler](#) APIs.

- **Debugging improvements.** The following new unmanaged debugging APIs provide better integration with a profiler. You can now access metadata inserted by the profiler as well as local variables and code produced by compiler ReJIT requests when dump debugging.
 - [SetWritableDatabaseUpdateMode Method](#)
 - [EnumerateLocalVariablesEx Method](#)
 - [GetLocalVariableEx Method](#)
 - [GetCodeEx Method](#)
 - [GetActiveReJitRequestILCode Method](#)
 - [GetInstrumentedILMap Method](#)
- **Event tracing changes.** .NET Framework 4.5.2 enables out-of-process, Event Tracing for Windows (ETW)-based activity tracing for a larger surface area. This enables Advanced Power Management (APM) vendors to provide lightweight tools that accurately track the costs of individual requests and activities that cross threads. These events are raised only when ETW controllers enable them; therefore,

the changes don't affect previously written ETW code or code that runs with ETW disabled.

- **Promoting a transaction and converting it to a durable enlistment**

[Transaction.PromoteAndEnlistDurable](#) is a new API added to .NET Framework 4.5.2 and 4.6:

C#

```
[System.Security.Permissions.PermissionSetAttribute(System.Security.Per  
missions.SecurityAction.LinkDemand, Name = "FullTrust")]  
public Enlistment PromoteAndEnlistDurable(Guid  
resourceManagerIdentifier,  
  
IPromotableSinglePhaseNotification promotableNotification,  
ISinglePhaseNotification  
enlistmentNotification,  
EnlistmentOptions  
enlistmentOptions)
```

The method may be used by an enlistment that was previously created by [Transaction.EnlistPromotableSinglePhase](#) in response to the [ITransactionPromoter.Promote](#) method. It asks [System.Transactions](#) to promote the transaction to an MSDTC transaction and to "convert" the promotable enlistment to a durable enlistment. After this method completes successfully, the [IPromotableSinglePhaseNotification](#) interface will no longer be referenced by [System.Transactions](#), and any future notifications will arrive on the provided [ISinglePhaseNotification](#) interface. The enlistment in question must act as a durable enlistment, supporting transaction logging and recovery. Refer to [Transaction.EnlistDurable](#) for details. In addition, the enlistment must support [ISinglePhaseNotification](#). This method can *only* be called while processing an [ITransactionPromoter.Promote](#) call. If that is not the case, a [TransactionException](#) exception is thrown.

What's new in .NET Framework 4.5.1

April 2014 updates:

- [Visual Studio 2013 Update 2](#) includes updates to the Portable Class Library templates to support these scenarios:
 - You can use Windows Runtime APIs in portable libraries that target Windows 8.1, Windows Phone 8.1, and Windows Phone Silverlight 8.1.

- You can include XAML (Windows.UI.Xaml types) in portable libraries when you target Windows 8.1 or Windows Phone 8.1. The following XAML templates are supported: Blank Page, Resource Dictionary, Templated Control, and User Control.
- You can create a portable Windows Runtime component (.winmd file) for use in Store apps that target Windows 8.1 and Windows Phone 8.1.
- You can retarget a Windows Store or Windows Phone Store class library like a Portable Class Library.

For more information about these changes, see [Portable Class Library](#).

- The .NET Framework content set now includes documentation for .NET Native, which is a precompilation technology for building and deploying Windows apps. .NET Native compiles your apps directly to native code, rather than to intermediate language (IL), for better performance. For details, see [Compiling Apps with .NET Native](#).
- The [.NET Framework Reference Source](#) provides a new browsing experience and enhanced functionality. You can now browse through the .NET Framework source code online, [download the reference](#) for offline viewing, and step through the sources (including patches and updates) during debugging. For more information, see the blog entry [A new look for .NET Reference Source](#).

New features and enhancements in the base classes in .NET Framework 4.5.1 include:

- Automatic binding redirection for assemblies. Starting with Visual Studio 2013, when you compile an app that targets .NET Framework 4.5.1, binding redirects may be added to the app configuration file if your app or its components reference multiple versions of the same assembly. You can also enable this feature for projects that target older versions of .NET Framework. For more information, see [How to: Enable and Disable Automatic Binding Redirection](#).
- Ability to collect diagnostics information to help developers improve the performance of server and cloud applications. For more information, see the [WriteEventWithRelatedActivityId](#) and [WriteEventWithRelatedActivityIdCore](#) methods in the [EventSource](#) class.
- Ability to explicitly compact the large object heap (LOH) during garbage collection. For more information, see the [GCSettings.LargeObjectHeapCompactionMode](#) property.

- Additional performance improvements such as ASP.NET app suspension, multi-core JIT improvements, and faster app startup after a .NET Framework update. For details, see the [.NET Framework 4.5.1 announcement](#) and the [ASP.NET app suspend](#) blog post.

Improvements to Windows Forms include:

- Resizing in Windows Forms controls. You can use the system DPI setting to resize components of controls (for example, the icons that appear in a property grid) by opting in with an entry in the application configuration file (app.config) for your app. This feature is currently supported in the following Windows Forms controls:
 - [PropertyGrid](#)
 - [TreeView](#)
 - Some aspects of the [DataGridView](#) (see [new features in 4.5.2](#) for additional controls supported)

To enable this feature, add a new <appSettings> element to the configuration file (app.config) and set the `EnableWindowsFormsHighDpiAutoResizing` element to `true`:

XML

```
<appSettings>
  <add key="EnableWindowsFormsHighDpiAutoResizing" value="true" />
</appSettings>
```

Improvements when debugging your .NET Framework apps in Visual Studio 2013 include:

- Return values in the Visual Studio debugger. When you debug a managed app in Visual Studio 2013, the Autos window displays return types and values for methods. This information is available for desktop, Windows Store, and Windows Phone apps. For more information, see [Examine return values of method calls](#).
- Edit and Continue for 64-bit apps. Visual Studio 2013 supports the Edit and Continue feature for 64-bit managed apps for desktop, Windows Store, and Windows Phone. The existing limitations remain in effect for both 32-bit and 64-bit apps (see the last section of the [Supported Code Changes \(C#\)](#) article).
- Async-aware debugging. To make it easier to debug asynchronous apps in Visual Studio 2013, the call stack hides the infrastructure code provided by compilers to support asynchronous programming, and also chains in logical parent frames so you can follow logical program execution more clearly. A Tasks window replaces the Parallel Tasks window and displays tasks that relate to a particular breakpoint, and also displays any other tasks that are currently active or scheduled in the app.

You can read about this feature in the "Async-aware debugging" section of the [.NET Framework 4.5.1 announcement](#).

- Better exception support for Windows Runtime components. In Windows 8.1, exceptions that arise from Windows Store apps preserve information about the error that caused the exception, even across language boundaries. You can read about this feature in the "Windows Store app development" section of the [.NET Framework 4.5.1 announcement](#).

Starting with Visual Studio 2013, you can use the [Managed Profile Guided Optimization Tool \(Mpg0.exe\)](#) to optimize Windows 8.x Store apps as well as desktop apps.

For new features in ASP.NET 4.5.1, see [ASP.NET and Web Tools for Visual Studio 2013 Release Notes](#).

What's new in .NET Framework 4.5

Base classes

- Ability to reduce system restarts by detecting and closing .NET Framework 4 applications during deployment. See [Reducing System Restarts During .NET Framework 4.5 Installations](#).
- Support for arrays that are larger than 2 gigabytes (GB) on 64-bit platforms. This feature can be enabled in the application configuration file. See the [`<gcAllowVeryLargeObjects>` element](#), which also lists other restrictions on object size and array size.
- Better performance through background garbage collection for servers. When you use server garbage collection in .NET Framework 4.5, background garbage collection is automatically enabled. See the [Background Server Garbage Collection](#) section of the [Fundamentals of Garbage Collection](#) topic.
- Background just-in-time (JIT) compilation, which is optionally available on multi-core processors to improve application performance. See [ProfileOptimization](#).
- Ability to limit how long the regular expression engine will attempt to resolve a regular expression before it times out. See the [Regex.MatchTimeout](#) property.
- Ability to define the default culture for an application domain. See the [CultureInfo](#) class.
- Console support for Unicode (UTF-16) encoding. See the [Console](#) class.

- Support for versioning of cultural string ordering and comparison data. See the [SortVersion](#) class.
- Better performance when retrieving resources. See [Package and deploy resources](#).
- Zip compression improvements to reduce the size of a compressed file. See the [System.IO.Compression](#) namespace.
- Ability to customize a reflection context to override default reflection behavior through the [CustomReflectionContext](#) class.
- Support for the 2008 version of the Internationalized Domain Names in Applications (IDNA) standard when the [System.Globalization.IdnMapping](#) class is used on Windows 8.
- Delegation of string comparison to the operating system, which implements Unicode 6.0, when the .NET Framework is used on Windows 8. When running on other platforms, the .NET Framework includes its own string comparison data, which implements Unicode 5.x. See the [String](#) class and the Remarks section of the [SortVersion](#) class.
- Ability to compute the hash codes for strings on a per application domain basis. See [`<UseRandomizedStringHashAlgorithm>` Element](#).
- Type reflection support split between [Type](#) and [TypeInfo](#) classes. See [Reflection in the .NET Framework for Windows Store Apps](#).

Managed Extensibility Framework (MEF)

In .NET Framework 4.5, the Managed Extensibility Framework (MEF) provides the following new features:

- Support for generic types.
- Convention-based programming model that enables you to create parts based on naming conventions rather than attributes.
- Multiple scopes.
- A subset of MEF that you can use when you create Windows 8.x Store apps. This subset is available as a [downloadable package](#) from the NuGet Gallery. To install the package, open your project in Visual Studio, choose **Manage NuGet Packages** from the **Project** menu, and search online for the [Microsoft.Composition](#) package.

For more information, see [Managed Extensibility Framework \(MEF\)](#).

Asynchronous file operations

In .NET Framework 4.5, new asynchronous features were added to the C# and Visual Basic languages. These features add a task-based model for performing asynchronous operations. To use this new model, use the asynchronous methods in the I/O classes. See [Asynchronous File I/O](#).

Tools

In .NET Framework 4.5, Resource File Generator (Resgen.exe) enables you to create a .resw file for use in Windows 8.x Store apps from a .resources file embedded in a .NET Framework assembly. For more information, see [Resgen.exe \(Resource File Generator\)](#).

Managed Profile Guided Optimization (Mpgc.exe) enables you to improve application startup time, memory utilization (working set size), and throughput by optimizing native image assemblies. The command-line tool generates profile data for native image application assemblies. See [Mpgc.exe \(Managed Profile Guided Optimization Tool\)](#). Starting with Visual Studio 2013, you can use Mpgc.exe to optimize Windows 8.x Store apps as well as desktop apps.

Parallel computing

.NET Framework 4.5 provides several new features and improvements for parallel computing. These include improved performance, increased control, improved support for asynchronous programming, a new dataflow library, and improved support for parallel debugging and performance analysis. See the entry [What's New for Parallelism in .NET Framework 4.5](#) in the Parallel Programming with .NET blog.

Web

ASP.NET 4.5 and 4.5.1 add model binding for Web Forms, WebSocket support, asynchronous handlers, performance enhancements, and many other features. For more information, see the following resources:

- [ASP.NET 4.5 and Visual Studio 2012](#)
- [ASP.NET and Web Tools for Visual Studio 2013 Release Notes](#)

Networking

.NET Framework 4.5 provides a new programming interface for HTTP applications. For more information, see the new [System.Net.Http](#) and [System.Net.Http.Headers](#) namespaces.

Support is also included for a new programming interface for accepting and interacting with a WebSocket connection by using the existing [HttpListener](#) and related classes. For more information, see the new [System.Net.WebSockets](#) namespace and the [HttpListener](#) class.

In addition, .NET Framework 4.5 includes the following networking improvements:

- RFC-compliant URI support. For more information, see [Uri](#) and related classes.
- Support for Internationalized Domain Name (IDN) parsing. For more information, see [Uri](#) and related classes.
- Support for Email Address Internationalization (EAI). For more information, see the [System.Net.Mail](#) namespace.
- Improved IPv6 support. For more information, see the [System.Net.NetworkInformation](#) namespace.
- Dual-mode socket support. For more information, see the [Socket](#) and [TcpListener](#) classes.

Windows Presentation Foundation (WPF)

In .NET Framework 4.5, Windows Presentation Foundation (WPF) contains changes and improvements in the following areas:

- The new [Ribbon](#) control, which enables you to implement a ribbon user interface that hosts a Quick Access Toolbar, Application Menu, and tabs.
- The new [INotifyDataErrorInfo](#) interface, which supports synchronous and asynchronous data validation.
- New features for the [VirtualizingPanel](#) and [Dispatcher](#) classes.
- Improved performance when displaying large sets of grouped data, and by accessing collections on non-UI threads.
- Data binding to static properties, data binding to custom types that implement the [ICustomTypeProvider](#) interface, and retrieval of data binding information from a binding expression.

- Repositioning of data as the values change (live shaping).
- Ability to check whether the data context for an item container is disconnected.
- Ability to set the amount of time that should elapse between property changes and data source updates.
- Improved support for implementing weak event patterns. Also, events can now accept markup extensions.

Windows Communication Foundation (WCF)

In .NET Framework 4.5, the following features have been added to make it simpler to write and maintain Windows Communication Foundation (WCF) applications:

- Simplification of generated configuration files.
- Support for contract-first development.
- Ability to configure ASP.NET compatibility mode more easily.
- Changes in default transport property values to reduce the likelihood that you will have to set them.
- Updates to the [XmlDictionaryReaderQuotas](#) class to reduce the likelihood that you will have to manually configure quotas for XML dictionary readers.
- Validation of WCF configuration files by Visual Studio as part of the build process, so you can detect configuration errors before you run your application.
- New asynchronous streaming support.
- New HTTPS protocol mapping to make it easier to expose an endpoint over HTTPS with Internet Information Services (IIS).
- Ability to generate metadata in a single WSDL document by appending `?singleWS` to the service URL.
- Websockets support to enable true bidirectional communication over ports 80 and 443 with performance characteristics similar to the TCP transport.
- Support for configuring services in code.
- XML Editor tooltips.
- [ChannelFactory](#) caching support.

- Binary encoder compression support.
- Support for a UDP transport that enables developers to write services that use "fire and forget" messaging. A client sends a message to a service and expects no response from the service.
- Ability to support multiple authentication modes on a single WCF endpoint when using the HTTP transport and transport security.
- Support for WCF services that use internationalized domain names (IDNs).

For more information, see [What's New in Windows Communication Foundation](#).

Windows Workflow Foundation (WF)

In .NET Framework 4.5, several new features were added to Windows Workflow Foundation (WF), including:

- State machine workflows, which were first introduced as part of .NET Framework 4.0.1 ([.NET Framework 4 Platform Update 1](#)). This update included several new classes and activities that enabled developers to create state machine workflows. These classes and activities were updated for .NET Framework 4.5 to include:
 - The ability to set breakpoints on states.
 - The ability to copy and paste transitions in the workflow designer.
 - Designer support for shared trigger transition creation.
 - Activities for creating state machine workflows, including: [StateMachine](#), [State](#), and [Transition](#).
- Enhanced Workflow Designer features such as the following:
 - Enhanced workflow search capabilities in Visual Studio, including [Quick Find](#) and [Find in Files](#).
 - Ability to automatically create a Sequence activity when a second child activity is added to a container activity, and to include both activities in the Sequence activity.
 - Panning support, which enables the visible portion of a workflow to be changed without using the scroll bars.
 - A new [Document Outline](#) view that shows the components of a workflow in a tree-style outline view and lets you select a component in the [Document](#)

Outline view.

- Ability to add annotations to activities.
- Ability to define and consume activity delegates by using the workflow designer.
- Auto-connect and auto-insert for activities and transitions in state machine and flowchart workflows.
- Storage of the view state information for a workflow in a single element in the XAML file, so you can easily locate and edit the view state information.
- A NoPersistScope container activity to prevent child activities from persisting.
- Support for C# expressions:
 - Workflow projects that use Visual Basic will use Visual Basic expressions, and C# workflow projects will use C# expressions.
 - C# workflow projects that were created in Visual Studio 2010 and that have Visual Basic expressions are compatible with C# workflow projects that use C# expressions.
- Versioning enhancements:
 - The new [WorkflowIdentity](#) class, which provides a mapping between a persisted workflow instance and its workflow definition.
 - Side-by-side execution of multiple workflow versions in the same host, including [WorkflowServiceHost](#).
 - In Dynamic Update, the ability to modify the definition of a persisted workflow instance.
- Contract-first workflow service development, which provides support for automatically generating activities to match an existing service contract.

For more information, see [What's New in Windows Workflow Foundation](#).

.NET for Windows 8.x Store apps

Windows 8.x Store apps are designed for specific form factors and leverage the power of the Windows operating system. A subset of .NET Framework 4.5 or 4.5.1 is available for building Windows 8.x Store apps for Windows by using C# or Visual Basic. This subset is called .NET for Windows 8.x Store apps and is discussed in an [overview](#).

Portable Class Libraries

The Portable Class Library project in Visual Studio 2012 (and later versions) enables you to write and build managed assemblies that work on multiple .NET Framework platforms. Using a Portable Class Library project, you choose the platforms (such as Windows Phone and .NET for Windows 8.x Store apps) to target. The available types and members in your project are automatically restricted to the common types and members across these platforms. For more information, see [Portable Class Library](#).

See also

- [The .NET Framework and Out-of-Band Releases](#)
- [What's new in accessibility in .NET Framework](#)
- [What's New in Visual Studio 2019](#)
- [ASP.NET](#)
- [What's New for C++ in Visual Studio](#)
- [Download the .NET SDK ↗](#)

Collaborate with us on GitHub

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).



.NET feedback

.NET is an open source project. Select a link to provide feedback:

 [Open a documentation issue](#)

 [Provide product feedback](#)

What's new in accessibility in .NET Framework

Article • 10/13/2022

.NET Framework aims to make applications more accessible for your users. Accessibility features allow an application to provide an appropriate experience for users of Assistive Technology. Starting with .NET Framework 4.7.1, .NET Framework includes a large number of accessibility improvements that allow developers to create accessible applications.

Accessibility switches

You can configure your app to opt into accessibility features if it targets .NET Framework 4.7 or an earlier version but is running on .NET Framework 4.7.1 or later. You can also configure your app to use legacy features (and not take advantage of accessibility features) if it targets .NET Framework 4.7.1 or later. Each .NET Framework version that includes accessibility features has a version-specific accessibility switch, which you add to the `<AppContextSwitchOverrides>` element in the `<runtime>` section of the application's configuration file. The following are the supported switches:

[+] Expand table

Version	Switch
.NET Framework 4.7.1	"Switch.UseLegacyAccessibilityFeatures"
.NET Framework 4.7.2	"Switch.UseLegacyAccessibilityFeatures.2"
.NET Framework 4.8	"Switch.UseLegacyAccessibilityFeatures.3"
August 11, 2020-KB4569746 Cumulative Update for .NET Framework 4.8	"Switch.UseLegacyAccessibilityFeatures.4"
.NET Framework 4.8.1	"Switch.UseLegacyAccessibilityFeatures.5"

Taking advantage of accessibility enhancements

The new accessibility features are enabled by default for applications that target .NET Framework 4.7.1 or later. In addition, applications that target an earlier version of the .NET Framework but are running on .NET Framework 4.7.1 or later can opt out of legacy accessibility behaviors (and thereby take advantage of accessibility improvements) by

adding switches to the `<AppContextSwitchOverrides>` element in the `<runtime>` section of the application's configuration file and setting their value to `false`. The following snippet shows how to opt in to the accessibility enhancements that were introduced in .NET Framework 4.7.1:

XML

```
<runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
key1=true|false;key2=true|false -->
    <AppContextSwitchOverrides
        value="Switch.UseLegacyAccessibilityFeatures=false" />
</runtime>
```

If you choose to opt in to accessibility features in a later .NET Framework version, you must also explicitly opt in to the features from earlier versions. To configure your app to take advantage of accessibility improvements in both .NET Framework 4.7.1 and 4.7.2, add the following `<AppContextSwitchOverrides>` element:

XML

```
<runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
key1=true|false;key2=true|false -->
    <AppContextSwitchOverrides
        value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi
lityFeatures.2=false" />
</runtime>
```

To configure your app to take advantage of accessibility improvements in .NET Framework 4.7.1, 4.7.2, 4.8, and the August 2020 cumulative update for .NET Framework 4.8, add the following `<AppContextSwitchOverrides>` element:

XML

```
<runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
key1=true|false;key2=true|false -->
    <AppContextSwitchOverrides
        value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi
lityFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false;Switch.Us
eLegacyAccessibilityFeatures.4=false" />
</runtime>
```

Restoring legacy behavior

Applications that target versions of .NET Framework starting with 4.7.1 can disable accessibility features by adding switches to the `<AppContextSwitchOverrides>` element in the `<runtime>` section of the application's configuration file and setting their value to `true`. For example, the following configuration opts out of accessibility features introduced in .NET Framework 4.7.2:

```
XML

<runtime>
    <!-- AppContextSwitchOverrides value attribute is in the form of
key1=true|false;key2=true|false -->
    <AppContextSwitchOverrides
        value="Switch.UseLegacyAccessibilityFeatures.2=true" />
</runtime>
```

What's new in accessibility in .NET Framework 4.8.1

.NET Framework 4.8.1 includes new accessibility features in the following areas:

- [Windows Forms](#)
- [Windows Presentation Foundation \(WPF\)](#)

Windows Forms

Added and improved UIA representations

Prior to .NET Framework 4.8.1, Windows Forms was missing support for a range of UIA patterns to support assistive technology to interact with applications. This deficiency could cause Narrator or other screen readers to report incomplete or incorrect information. It could also affect important functionality like moving a cursor through the text in a [TextBox](#) control. With .NET Framework 4.8.1, all of the required patterns for the common controls have been implemented. This new functionality gives users of assistive technology a much richer application interaction experience.

- Added support for the UIA expand/collapse pattern to the [DateTimePicker](#) control.
- Added UIA support to the [MonthCalendar](#) control. Now assistive technology tools such as Narrator can navigate through the individual dates in the control.
- Implemented text pattern support for all text-based controls, including the [TextBox](#), [MaskedTextBox](#), [PropertyGrid](#) edit control, [DataGridViewTextBoxCell](#), [ToolStripTextBox](#), and [DomainUpDown](#) controls.

- [ToolTip](#) now follows [WCAG2.1 guidelines](#) to be persistent, dismissable, and hoverable on Windows 11. Changes to tooltip behavior are limited to Windows 11 systems that have .NET Framework 4.8.1 installed, and only apply to applications where a timeout was not set for the tooltip. Tooltips that are persisting can be dismissed with either the Esc key or the Ctrl key or by navigating to a control with another tooltip set.

Various bug fixes for existing accessibility features

- Narrator can now focus on an empty [DataGridView](#) control.
- Addressed an issue that caused screen readers to count hidden columns when announcing the column count in a [DataGridView](#) control.
- Addressed an issue that caused the [DataGridView](#) to ignore the font settings set in the `DataGridviewCellStyle` if the underlying form had a `Font` property that differed from `DefaultFont`.
- Updated the `AccessibleName` property of the [DataGridView](#) control's internal scroll bars to remove the text "ScrollBar".
- Fixed the color of a `DataGridViewLinkCell` when the cell is selected.
- Fixed an issue with custom [DataGridView](#) controls where there was no `ControlType` or `LocalizedControlType` provided for custom [DataGridViewCell](#) elements.
- Updated the luminosity ratio to 3.5:1 for [ToolStripButton](#) controls that have `ToolStripRenderMode` set to `System`.
- Improved keyboard navigation in a [ToolStrip](#) when the element is a `ToolStripComboBox` type.
- Updated the background color of the [ToolStripButton](#) in high contrast mode.
- Ensured that there is a bounding rectangle reported to assistive technology for the [ToolStripSeparator](#).
- Fixed an issue that could cause the screen reader JAWS to crash when reading the [PropertyGrid](#) control.
- Ensured that the UIA hierarchy tree for a [PropertyGrid](#) control is updated when a complex entry like `Font` is expanded. Also ensured that the tree is updated properly when the entry is then collapsed and no longer visible.
- [PropertyGrid](#) categories now have a localized control type of `PropertyGrid` `category`.
- Fixed an issue with the [ComboBox](#) that could cause the app to crash under Accessibility Insights for Windows.
- Updated the border color for the [Button](#) to have more contrast in the default colors.
- Enabled assistive technology tools to access the `ControlBox` buttons of a maximized MDI child form.

- The `AccessibleName` property of a `DomainUpDown` control has a new default value of an empty string. The empty string will encourage developers to create a meaningful name rather than accepting the previous non-empty default value.
- Updated the `AccessibleName` property of the Print button in the `PrintPreviewDialog` from "Print Button" to "Print" to avoid redundancy when screen readers announce the control and its type.
- Updated UIA list controls to remove an empty list element when a `PropertyGrid` cell of type `ComboBox` is closed and no longer visible.

Windows Presentation Foundation (WPF)

Accessible Tooltip handling improvements

In this release, WPF improved the experience by ensuring that a tooltip in the current window can easily be dismissed by using the Esc key, the Ctrl key (by itself), or by the combination Ctrl+Shift+F10. The scope of the Esc key was reduced in this release to apply only to the current window, when previously it would have applied to any open tooltip in the application. By default, WPF tooltips will follow [WCAG2.1 guidelines](#) to be persistent, dismissable, and hoverable.

What's new in accessibility in the August 11, 2020 Cumulative Update for .NET Framework 4.8

The August 11, 2020-KB4569746 Cumulative Update for .NET Framework 4.8 includes new accessibility features in Windows Forms:

- Addresses an issue with announcing `PropertyGrid` control items and a category's expanded/collapsed state by screen readers.
- Updates the accessible patterns of the `PropertyGrid` control and its inner elements.
- Updates the accessible names of the `PropertyGrid` control inner elements so they're correctly announced by screen readers.
- Addresses bounding rectangle accessible properties for the `PropertyGridView` controls.

- Enables screen readers to correctly announce the expanded/collapsed state of `DataGridView` combo box cells.

What's new in accessibility in .NET Framework 4.8

.NET Framework 4.8 includes new accessibility features in the following areas:

- [Windows Forms](#)
- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Workflow Foundation \(WF\) workflow designer](#)

Windows Forms

In .NET Framework 4.8, Windows Forms adds support for LiveRegions and Notification Events to many commonly used controls. It also adds support for ToolTips when a user navigates to a control by using the keyboard.

UIA LiveRegions Support in Labels and StatusStrips

UIA LiveRegions allow application developers to notify screen readers of a text change in a control that is located apart from the location where the user is working. This is useful, for example, for a [StatusStrip](#) control that shows a connection status. If the connection is dropped and the status changes, the developer might want to notify the screen reader.

Starting with .NET Framework 4.8, Windows Forms implements UIA LiveRegions for both the [Label](#) and [StatusStrip](#) controls. For example, the following code uses the LiveRegion in a [Label](#) control named `label1`:

C#

```
public Form1()
{
    InitializeComponent();
    label1.AutomationLiveSetting = AutomationLiveSetting.Polite;
}

...
Label1.Text = "Ready!";
```

Narrator announces “Ready” regardless of where the user is interacting with the application.

You can also implement your [UserControl](#) as a LiveRegion:

C#

```
using System;
using System.Windows.Forms;
using System.Windows.Forms.Automation;

namespace WindowsFormsApplication
{
    public partial class UserControl1 : UserControl, IAutomationLiveRegion
    {
        public UserControl1()
        {
            InitializeComponent();
        }

        public AutomationLiveSetting AutomationLiveSetting { get; set; }
        private AutomationLiveSetting IAutomationLiveRegion.GetLiveSetting()
        {
            return this.AutomationLiveSetting;
        }

        protected override void OnTextChanged(EventArgs e)
        {
            base.OnTextChanged(e);

            AutomationNotifications.UiaRaiseLiveRegionChangedEvent(this.AccessibleObject);
        }
    }
}
```

UIA notification events

The UIA Notification event, introduced in Windows 10 Fall Creators Update, allows your app to raise a UIA event, which leads to Narrator simply making an announcement based on text you supply with the event, without the need to have a corresponding control in the UI. In some scenarios, this is a straightforward way to dramatically improve the accessibility of your app. It can also be useful to notify of the progress of some process that may take a long time. For more information about UIA Notification Events, see [Can your desktop app leverage the new UI Notification event?](#).

The following example raises the [Notification event](#):

C#

```
MethodInfo raiseMethod =
typeof(AccessibleObject).GetMethod("RaiseAutomationNotification");
if (raiseMethod != null) {
    raiseMethod.Invoke(progressBar1.AccessibilityObject, new object[3]
{ /*Other*/ 4, /*All*/ 2, "The progress is 50%." });
}
```

ToolTips on keyboard access

In applications that target .NET Framework 4.7.2 and earlier versions, a control [tooltip](#) can only be triggered to pop up by moving a mouse pointer into the control. Starting with .NET Framework 4.8, a keyboard user can trigger a control's tooltip by focusing the control using a Tab key or arrow keys with or without modifier keys. This particular accessibility enhancement requires an additional [AppContext switch](#):

XML

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1"/>
    </startup>
    <runtime>
        <!-- AppContextSwitchOverrides values are in the form of
key1=true|false;key2=true|false -->
        <!-- Please note that disabling Switch.UseLegacyAccessibilityFeatures,
Switch.UseLegacyAccessibilityFeatures.2 and
Switch.UseLegacyAccessibilityFeatures.3 is required to disable
Switch.System.Windows.Forms.UseLegacyToolTipDisplay -->
        <AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibi
lityFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false;Switch.Sy
stem.Windows.Forms.UseLegacyToolTipDisplay=false"/>
    </runtime>
</configuration>
```

The following figure shows the tooltip when the user has selected a button with the keyboard.



Windows Presentation Foundation (WPF)

Starting with .NET Framework 4.8, WPF includes a number of accessibility improvements.

Screen narrators no longer announce elements with Collapsed or Hidden visibility

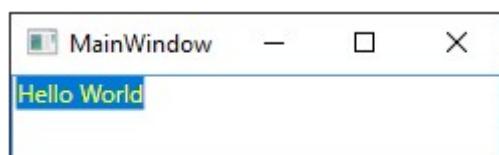
Elements with collapsed or hidden visibility are no longer announced by screen reader. User interfaces that contain elements with a Visibility of [Visibility.Collapsed](#) or [Visibility.Hidden](#) can be misrepresented by screen readers if they are announced to the user. Starting with .NET Framework 4.8, WPF no longer includes collapsed or hidden elements in the Control View of the UIAutomation tree, so the screen readers can no longer announce these elements.

SelectionTextBrush property for use with non-Adorner based text selection

In .NET Framework 4.7.2, WPF added the ability to draw [TextBox](#) and [PasswordBox](#) text selection without using the Adorner layer. The foreground color of the selected text in this scenario was dictated by [SystemColors.HighlightTextBrush](#).

.NET Framework 4.8 adds a new property, `SelectionTextBrush`, that allows developers to select the specific brush for the selected text when using non-Adorner based text selection. This property works only on [TextBoxBase](#)-derived controls and the [PasswordBox](#) control in WPF applications with non-Adorner-based text selection enabled. It does not work on the [RichTextBox](#) control. If non-Adorner-based text selection is not enabled, this property is ignored.

To use this property, simply add it to your XAML code and use the appropriate brush or binding. The resulting text selection looks like this:



You can combine the use of the `SelectionBrush` and `SelectionTextBrush` properties to generate any background and foreground color combination that you deem appropriate.

Support for the UIAutomation ControllerFor property

UIAutomation's `ControllerFor` property returns an array of automation elements that are manipulated by the automation element that supports this property. This property is commonly used for Auto-suggest accessibility. `ControllerFor` is used when an automation element affects one or more segments of the application UI or the desktop. Otherwise, it is hard to associate the impact of the control operation with UI elements. This feature adds the ability for controls to provide a value for the `ControllerFor` property.

.NET Framework 4.8 adds a new virtual method, `GetControlledPeersCore()`. To provide a value for the `ControllerFor` property, simply override this method and return a `List<AutomationPeer>` for the controls being manipulated by this `AutomationPeer`:

C#

```
public class AutoSuggestTextBox : TextBox
{
    protected override AutomationPeer OnCreateAutomationPeer()
    {
        return new AutoSuggestTextBoxAutomationPeer(this);
    }

    public ListBox SuggestionListBox;
}

internal class AutoSuggestTextBoxAutomationPeer : TextBoxAutomationPeer
{
    public AutoSuggestTextBoxAutomationPeer(AutoSuggestTextBox owner) :
    base(owner)
    {

    }

    protected override List<AutomationPeer> GetControlledPeersCore()
    {
        List<AutomationPeer> controlledPeers = new List<AutomationPeer>();
        AutoSuggestTextBox owner = Owner as AutoSuggestTextBox;

        controlledPeers.Add(UIElementAutomationPeer.CreatePeerForElement(owner.Sugge
stionListBox));
        return controlledPeers;
    }
}
```

Tooltips on keyboard access

In .NET Framework 4.7.2 and earlier versions, tooltips display only when the user hovers the mouse cursor over a control. In .NET Framework 4.8, tooltips also display on keyboard focus, as well as via a keyboard shortcut.

To enable this feature, an application needs to target .NET Framework 4.8 or opt-in by using the `Switch.UseLegacyAccessibilityFeatures.3` and `Switch.UseLegacyToolTipDisplay` `AppContext` switches. The following is a sample application configuration file:

```
XML

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <runtime>
        <AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibilityFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false;Switch.UseLegacyToolTipDisplay=false" />
    </runtime>
</configuration>
```

Once enabled, all controls that contain a tooltip display it once the control receives keyboard focus. The tooltip can be dismissed over time or when the keyboard focus changes. Users can also dismiss the tooltip manually by using a new keyboard shortcut, Ctrl+Shift+F10. Once the tooltip has been dismissed, it can be displayed again by using the same keyboard shortcut.

ⓘ Note

Ribbon tooltips on Ribbon controls won't show on keyboard focus; they only show via the keyboard shortcut.

Added Support for `SizeOfSet` and `PositionInSet` UIAutomation properties

Windows 10 introduced two new UIAutomation properties, `SizeOfSet` and `PositionInSet`, which are used by applications to describe the count of items in a set. UIAutomation client applications such as screen readers can then query an application for these properties and announce an accurate representation of the application's UI.

Starting with .NET Framework 4.8, WPF exposes these two properties to UIAutomation in WPF applications. This can be accomplished in two ways:

- By using dependency properties.

WPF adds two new dependency properties, `AutomationProperties.SizeOfSet` and `AutomationProperties.PositionInSet`. A developer can use XAML to set their values:

XAML

```
<Button AutomationProperties.SizeOfSet="3"
       AutomationProperties.PositionInSet="1">Button 1</Button>

<Button AutomationProperties.SizeOfSet="3"
       AutomationProperties.PositionInSet="2">Button 2</Button>

<Button AutomationProperties.SizeOfSet="3"
       AutomationProperties.PositionInSet="3">Button 3</Button>
```

- By overriding AutomationPeer virtual methods.

The `GetSizeOfSetCore()` and `GetPositionInSetCore()` virtual methods were added to the `AutomationPeer` class. A developer can provide values for `SizeOfSet` and `PositionInSet` by overriding these methods, as shown in the following example:

C#

```
public class MyButtonAutomationPeer : ButtonAutomationPeer
{
    protected override int GetSizeOfSetCore()
    {
        // Call into your own logic to provide a value for SizeOfSet
        return CalculateSizeOfSet();
    }

    protected override int GetPositionInSetCore()
    {
        // Call into your own logic to provide a value for PositionInSet
        return CalculatePositionInSet();
    }
}
```

In addition, items in `ItemsControl` instances provide a value for these properties automatically without additional action from the developer. If an `ItemsControl` is grouped, the collection of groups is represented as a set, and each group is counted as a separate set, with each item inside that group providing its position inside that group as well as the size of the group. Automatic values are not affected by virtualization. Even

if an item is not realized, it is still counted toward the total size of the set and affects the position in the set of its sibling items.

Automatic values are only provided if the application targets .NET Framework 4.8. For applications that target an earlier version of the .NET Framework, you can set the `Switch.UseLegacyAccessibilityFeatures.3` [AppContext switch](#), as shown in the following App.config file:

```
XML

<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>
    <runtime>
        <AppContextSwitchOverrides
value="Switch.UseLegacyAccessibilityFeatures=false;Switch.UseLegacyAccessibilityFeatures.2=false;Switch.UseLegacyAccessibilityFeatures.3=false" />
    </runtime>
</configuration>
```

Windows Workflow Foundation (WF) workflow designer

The workflow designer includes the following changes in .NET Framework 4.8:

- Users using Narrator will see improvements in FlowSwitch case labels.
- Users using Narrator will see improvements in button descriptions.
- Users who choose High Contrast themes will see improvements in the visibility of the Workflow Designer and its controls, like better contrast ratios between elements and more noticeable selection boxes used for focus elements.

If your application targets .NET Framework 4.7.2 or an earlier version, you can opt into these changes by setting the `Switch.UseLegacyAccessibilityFeatures.3` [AppContext switch](#) to `false` in your application configuration file. For more information, see the [Taking advantage of accessibility enhancements](#) section in this article.

What's new in accessibility in .NET Framework 4.7.2

.NET Framework 4.7.2 includes new accessibility features in the following areas:

- Windows Forms
- Windows Presentation Foundation (WPF)

Windows Forms

OS-defined colors in High Contrast themes

Starting with .NET Framework 4.7.2, Windows Forms uses colors defined by the operating system in High Contrast themes. This affects the following controls:

- The drop-down arrow of the [ToolStripDropDownButton](#) control.
- The [Button](#), [RadioButton](#) and [CheckBox](#) controls with [FlatStyle](#) set to [FlatStyle.Flat](#) or [FlatStyle.Popup](#). Previously, selected text and background colors were not contrasting and were hard to read.
- Controls contained within a [GroupBox](#) that has its [Enabled](#) property set to `false`.
- The [ToolStripButton](#), [ToolStripComboBox](#), and [ToolStripDropDownButton](#) controls, which have an increased luminosity contrast ratio in High Contrast Mode.
- The [LinkColor](#) property of the [DataGridViewLinkCell](#).

Narrator improvements

Starting with .NET Framework 4.7.2, Narrator support is enhanced as follows:

- It announces the value of the [ToolStripMenuItem.ShortcutKeys](#) property when announcing the text of a [ToolStripMenuItem](#).
- It indicates when a [ToolStripMenuItem](#) has its [Enabled](#) property set to `false`.
- It gives feedback on the state of a check box when the [ListView.CheckBoxes](#) property is set to `true`.
- Narrator's Scan Mode focus order is consistent with the visual order of the controls on the ClickOnce download dialog window.

DataGridView improvements

Starting with .NET Framework 4.7.2, the [DataGridView](#) control has introduced the following accessibility improvements:

- Rows can be sorted using the keyboard. A user can use the `F3` key in order to sort by the current column.

- The column width of the current cell can be increased or decreased with the `Alt` + `Left/Right arrow` keys.
- When the `DataGridView.SelectionMode` is set to `DataGridViewSelectionMode.FullRowSelect`, the column header changes color to indicate the current column as the user tabs through the cells in the current row.
- The `AccessibleObject.Parent` property of a `System.Windows.Forms.DataGridViewLinkCell.DataGridViewLinkCellAccessibleObject` returns the correct parent control.

Improved visual cues

- The `RadioButton` and `CheckBox` controls with an empty `Text` property display a focus indicator when they receive the focus.

Improved Property Grid Support

- The `PropertyGrid` control child elements now return a `true` for the `IsReadOnlyProperty` property only when a `PropertyGrid` element is enabled.
- The `PropertyGrid` control child elements return a `false` for the `IsEnabledProperty` property only when a `PropertyGrid` element can be changed by the user.

Improved keyboard navigation

- The `ToolStripButton` control allows focus when contained within a `ToolStripPanel` that has the `TabStop` property set to `true`

Windows Presentation Foundation (WPF)

Changes to the CheckBox and RadioButton controls

In .NET Framework 4.7.1 and earlier versions, the WPF `System.Windows.Controls.CheckBox` and `System.Windows.Controls.RadioButton` controls have inconsistent and, in Classic and High Contrast themes, incorrect focus visuals. These issues occur in cases where the controls do not have any content set. This can make the transition between themes confusing and the focus visual hard to see.

In .NET Framework 4.7.2, these visuals are now more consistent across themes and more easily visible in Classic and High Contrast themes.

WinForms controls hosted in a WPF application

For WinForms control hosted in a WPF application in .NET Framework 4.7.1 and earlier versions, users couldn't tab out of the WinForms layer if the first or last control in that layer is the WPF [ElementHost](#) control. In .NET Framework 4.7.2, users are now able to tab out of the WinForms layer.

However, automated applications that rely on focus never escaping the WinForms layer may no longer work as expected.

What's new in accessibility in .NET Framework 4.7.1

.NET Framework 4.7.1 includes new accessibility features in the following areas:

- [Windows Presentation Foundation \(WPF\)](#)
- [Windows Forms](#)
- [ASP.NET web controls](#)
- [.NET SDK Tools](#)
- [Windows Workflow Foundation \(WF\) Workflow Designer](#)

Windows Presentation Foundation (WPF)

Screen reader improvements

If accessibility improvements are enabled, .NET Framework 4.7.1 includes the following enhancements that affect screen readers:

- In .NET Framework 4.7 and earlier versions, [Expander](#) controls were announced by screen readers as buttons. Starting with .NET Framework 4.7.1, they are correctly announced as expandable/collapsible groups.
- In .NET Framework 4.7 and earlier versions, [DataGridCell](#) controls were announced by screen readers as "custom." Starting with .NET Framework 4.7.1, they are now correctly announced as data grid cell (localized).
- Starting with .NET Framework 4.7.1, screen readers announce the name of an editable [ComboBox](#).
- In .NET Framework 4.7 and earlier versions, [PasswordBox](#) controls were announced as "no item in view" or had otherwise incorrect behavior. This issue is fixed starting with .NET Framework 4.7.1.

UIAutomation LiveRegion support

Screen readers such as Narrator help people read the UI contents of an application, usually by text-to-speech output of the UI content that has the focus. However, if a UI element changes and does not have the focus, the user may not be notified and may miss important information. Live regions aim at solving this problem. A developer can use them to inform the screen reader or any other UIAutomation client that an important change has been made to a UI element. The screen reader can then decide how and when to inform the user of this change.

To support live regions, the following APIs have been added to WPF:

- The [AutomationElementIdentifiers.LiveSettingProperty](#) and [AutomationElementIdentifiers.LiveRegionChangedEvent](#) fields, which identify the **LiveSetting** property and the **LiveRegionChanged** event. They can be set by using XAML.
- The **AutomationProperties.LiveSetting** attached property, which informs the screen reader of the importance of the UI change to the user.
- The [AutomationProperties.LiveSettingProperty](#) property, which identifies the **AutomationProperties.LiveSetting** attached property.
- The [AutomationPeer.GetLiveSettingCore](#) method, which can be overridden to provide a **LiveSetting** value.
- The [AutomationProperties.GetLiveSetting](#) and [AutomationProperties.SetLiveSetting](#) methods, which get and set a **LiveSetting** value.
- The [System.Windows.Automation.AutomationLiveSetting](#) enumeration, which defines the following possible **LiveSetting** values:
 - [AutomationLiveSetting.Off](#). The element does not send notifications if the content of the live region has changed.
 - [AutomationLiveSetting.Polite](#). The element sends non-interruptive notifications if the content of the live region has changed.
 - [AutomationLiveSetting.Assertive](#). The element sends interruptive notifications if the content of the live region has changed.

You can create a LiveRegion by setting the **AutomationProperties.LiveSetting** property on the element of interest, as shown in the following example:

XAML

```
<TextBlock Name="myTextBlock"  
AutomationProperties.LiveSetting="Assertive">announcement</TextBlock>
```

When the data in the live region changes and you need to inform a screen reader, you explicitly raise an event, as shown in the following sample.

C#

```
var peer = FrameworkElementAutomationPeer.FromElement(myTextBlock);  
  
peer.RaiseAutomationEvent(AutomationEvents.LiveRegionChanged);
```

High contrast

Starting with .NET Framework 4.7.1, improvements in high contrast have been made to various WPF controls. They are now visible when the [HighContrast](#) theme is set. These include:

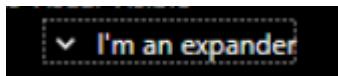
- [Expander](#) control

The focus visual for the [Expander](#) control is now visible. The keyboard visuals for [ComboBox](#), [ListBox](#), and [RadioButton](#) controls are visible as well. For example:

Before:



After:



- [CheckBox](#) and [RadioButton](#) controls

The text in the [CheckBox](#) and [RadioButton](#) controls is now easier to see when selected in high contrast themes. For example:

Before:



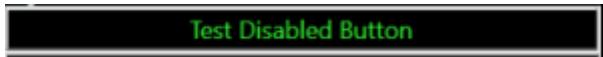
After:



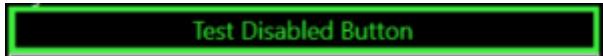
- [ComboBox](#) control

Starting with .NET Framework 4.7.1, the border of a disabled [ComboBox](#) control is the same color as disabled text. For example:

Before:



After:



In addition, disabled and focused buttons use the correct theme color.

Before:



After:



Finally, in .NET Framework 4.7 and earlier versions, setting a [ComboBox](#) control's style to `Toolbar.ComboBoxStyleKey` caused the drop-down arrow to be invisible. This issue is fixed starting with .NET Framework 4.7.1. For example:

Before:



After:



- [DataGrid](#) control

Starting with .NET Framework 4.7.1, the sort indicator arrow in [DataGrid](#) controls now uses correct theme colors. For example:

Before:

SomeText	SomeCheckbox
A	<input type="checkbox"/>
B	<input checked="" type="checkbox"/>

After:

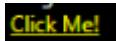


In addition, in .NET Framework 4.7 and earlier versions, the default link style changed to an incorrect color on mouse over in high contrast modes. This is resolved starting with .NET Framework 4.7.1. Similarly, [DataGrid](#) checkbox columns use the expected colors for keyboard focus feedback starting with .NET Framework 4.7.1.

Before:



After:



For more information on WPF accessibility improvements in .NET Framework 4.7.1, see [Accessibility improvements in WPF](#).

Windows Forms accessibility improvements

In .NET Framework 4.7.1, Windows Forms (WinForms) includes accessibility changes in the following areas.

Improved display in High Contrast mode

Starting with .NET Framework 4.7.1, various WinForms controls offer improved rendering in the HighContrast modes available in the operating system. Windows 10 has changed the values for some high contrast system colors, and Windows Forms is based on the Windows 10 Win32 framework. For the best experience, run on the latest version of Windows and opt in to the latest OS changes by adding an app.manifest file in a test application and uncomment the Windows 10 supported OS line so that it looks the following:

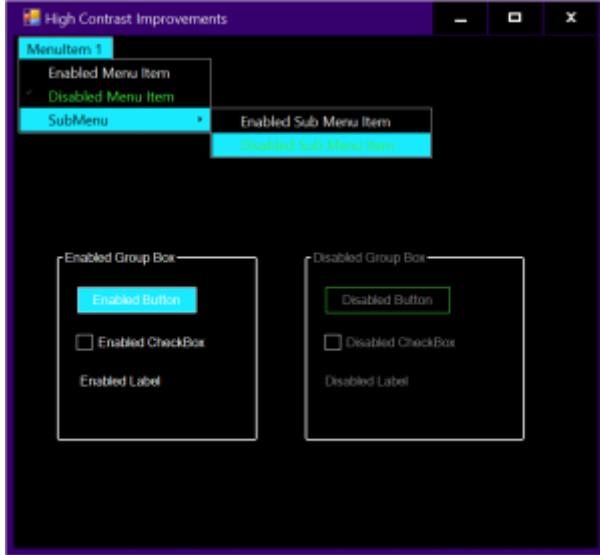
XML

```
<!-- Windows 10 -->
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}" />
```

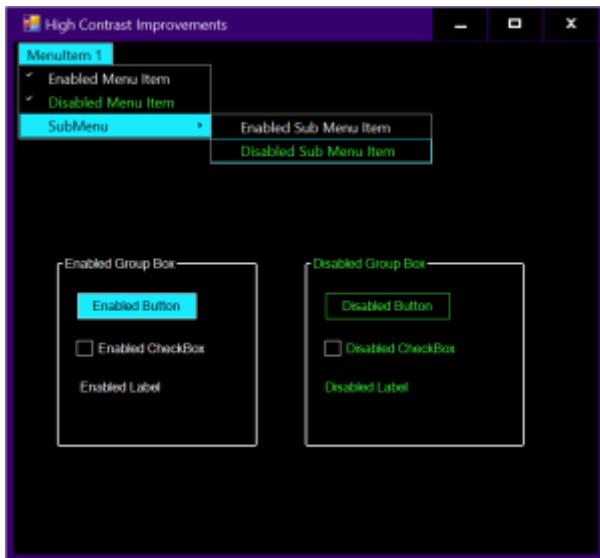
Some examples of high contrast changes include:

- Checkmarks in **MenuStrip** items are easier to view.
- When selected, disabled **MenuStrip** items are easier to view.
- Text in a selected **Button** control contrasts with the selection color.
- Disabled text is easier to read. For example:

Before:



After:



- High contrast improvements in the Thread Exception Dialog.

Improved Narrator support

Windows Forms in .NET Framework 4.7.1 includes the following accessibility improvements for the Narrator:

- The [MonthCalendar](#) control can be accessed by the Narrator, as well as by other UI automation tools.
- The [CheckedListBox](#) control notifies Narrator when an item's check state has changed so the user is notified that they've changed the value of a list item.
- The [DataGridViewCell](#) control reports the correct read-only status to Narrator.
- Narrator can now read disabled [ToolStripMenuItem](#) text, whereas previously it would skip over disabled menu items.

Enhanced support for UIAutomation accessibility patterns

Starting with .NET Framework 4.7.1, developers of accessibility technology tools can leverage common API accessibility patterns and properties for several WinForms controls. These accessibility improvements include:

- The [ComboBox](#) and [ToolStripSplitButton](#) now support the [expand/collapse pattern](#).
- The [DataGridViewCheckBoxCell](#) now supports the [toggle pattern](#).
- The [ToolStripItem](#) control supports the [Name](#) property and the [expand/collapse pattern](#).
- The [NumericUpDown](#) and [DomainUpDown](#) controls support the [Name](#) property.

Improved property browser experience

Starting with .NET Framework 4.7.1, Windows Forms includes:

- Better keyboard navigation through the various drop-down selection windows.
- A reduction of unnecessary tab stops.
- Better reporting of control types.
- Improved narrator behavior.

ASP.NET web controls

Starting with .NET Framework 4.7.1 and Visual Studio 2017 version 15.3, ASP.NET improves how ASP.NET web controls work with accessibility technology in Visual Studio. Changes include the following:

- Changes to implement missing UI accessibility patterns in controls, like the **Add Field** dialog in the **Details View** wizard, or the **Configure ListView** dialog of the **ListView** wizard.

- Changes to improve the display in High Contrast mode, like the **Data Pager Fields Editor**.
- Changes to improve the keyboard navigation experiences for controls, like the **Fields** dialog in the **Edit Pager Fields** wizard of the DataPager control, the **Configure ObjectContext** dialog, or the **Configure Data Selection** dialog of the **Configure Data Source** wizard.

.NET SDK Tools

The [Configuration Editor Tool \(SvcConfigEditor.exe\)](#) and [Service Trace Viewer Tool \(SvcTraceViewer.exe\)](#) have been improved by fixing varied accessibility issues. Most of these were small issues, like a name not being defined or certain UI automation patterns not being implemented correctly. While many users won't be aware of these incorrect values, customers who use assistive technologies like screen readers will find these SDK tools more accessible.

These enhancements change some previous behaviors, such as keyboard focus order.

Windows Workflow Foundation (WF) Workflow Designer

Accessibility changes in the Workflow Designer include the following:

- The tab order changes to left to right and top to bottom in some controls:
 - The initialize correlation window for setting correlation data for the [InitializeCorrelation](#) activity.
 - The content definition window for the [Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#) activities.
- More functions are available via the keyboard:
 - When editing the properties of an activity, property groups can be collapsed by keyboard the first time they are focused.
 - Warning icons are accessible by keyboard.
 - The **More Properties** button in the **Properties** window is accessible by keyboard.
 - Keyboard users can access the header items in the **Arguments** and **Variables** panes of the Workflow Designer.

- Improved visibility of items with focus, such as when:
 - Adding rows to data grids used by the Workflow Designer and activity designers.
 - Tabbing through fields in the [ReceiveReply](#) and [SendReply](#) activities.
 - Setting default values for variables or arguments
- Screen readers can now correctly recognize:
 - Breakpoints set in the workflow designer.
 - The [FlowSwitch<T>](#), [FlowDecision](#), and [CorrelationScope](#) activities.
 - The contents of the [Receive](#) activity.
 - The Target Type for the [InvokeMethod](#) activity.
 - The Exception combo box and the Finally section in the [TryCatch](#) activity.
 - The Message Type combo box, the splitter in the Add Correlation Initializers window, the Content Definition window, and the CorrelatesOn Definition window in the messaging activities ([Receive](#), [Send](#), [SendReply](#), and [ReceiveReply](#)).
 - State machine transitions and transitions destinations.
 - Annotations and connectors on [FlowDecision](#) activities.
 - The context (right-click) menus for activities.
 - The property value editors, the Clear Search button, the By Category and Alphabetical sort buttons, and the Expression Editor dialog in the properties grid.
 - The zoom percentage in the Workflow Designer.
 - The separator in [Parallel](#) and [Pick](#) activities.
 - The [InvokeDelegate](#) activity.
 - The Select Types window for dictionary activities
(`Microsoft.Activities.AddToDictionary< TKey, TValue >`,
`Microsoft.Activities.RemoveFromDictionary< TKey, TValue >`, etc.).
 - The Browse and Select .NET Type window.

- Breadcrumbs in the Workflow Designer.
- Users who choose High Contrast themes will see many improvements in the visibility of the Workflow Designer and its controls, like better contrast ratios between elements and more noticeable selection boxes used for focus elements.

See also

- [What's new in the .NET Framework](#)

 [Collaborate with us on GitHub](#)

The source for this content can be found on GitHub, where you can also create and review issues and pull requests. For more information, see [our contributor guide](#).

.NET

.NET feedback

.NET is an open source project. Select a link to provide feedback:

-  [Open a documentation issue](#)
-  [Provide product feedback](#)

Obsoletions in the .NET Framework class library

Article • 04/28/2023

.NET Framework changed over time. Each new version added new types and type members that provided new functionality. Existing types and their members also changed over time. For example, some types became less important as the technology they supported was replaced by a new technology, and some methods were superseded by newer methods that are superior in some way.

.NET Framework and the common language runtime strive to support backward compatibility (allowing applications that were developed with one version of .NET Framework to run on the next version of .NET Framework). This makes it difficult to simply remove a type or a type member. Instead, .NET Framework indicated that a type or a type member should no longer be used by marking it as *obsolete* or *deprecated*. By obsoleting a type or a member, developers were aware that it would go away and had time to respond to its removal. However, existing code that uses the type or member continued to run in the new version of .NET.

ⓘ Note

In .NET (Core), obsoleting an API doesn't necessarily mean that the API will be removed. For more information, see [API removal in .NET](#).

The `ObsoleteAttribute` attribute

.NET Framework indicates that a type or type member is obsolete by marking it with the [ObsoleteAttribute](#) attribute. Applying the attribute to a type or member indicates that type or member will be removed in some future version without breaking compiled code that uses that member.

In addition to indicating that a type or a type member is obsolete, [ObsoleteAttribute](#) defines how the compiler handles source code that includes that type or member. The compiler can compile the code but emit a warning message, or it can treat the use of the type or member as an error. In the first case, the code can successfully compile, but a warning message indicates that the type or member is obsolete. In the second case, compilation fails.

Even if compilation produces an error instead of a warning message, [ObsoleteAttribute](#) does not affect run-time behavior. That is, applications that use the type or member and that have compiled successfully will always run successfully. Only the attempt to recompile an application that uses the type or member fails.

How to handle obsolete types and members

When you upgrade and recompile existing code, using an obsolete type or member that produces a compiler warning in your application is acceptable. However, you should review the compiler warning message to determine whether you should change your application code. If the message does not point to a suitable alternative, you should do either of the following:

- Change your code by removing the use of the type or member, if possible.
-or-
- Review the documentation for this technology area to determine how to respond to the deprecation.

You may choose not to recompile existing code against a later version of .NET Framework. Instead, you can specify the version of .NET Framework against which your existing compiled code runs. For example, suppose you have an application named *app1.exe* that was compiled against .NET Framework 3.5, but you want the application to run against .NET Framework 4.5. This requires the following steps:

1. Create a configuration file for your main executable and name it *appName.exe.config*, where *appName* is the name of the application executable. For the application named *app1.exe* in our example, you would create a configuration file named *app1.exe.config*.
2. Add the following to the configuration file.

XML

```
<configuration>
  <startup>
    <supportedRuntime version="v4.0" />
  </startup>
</configuration>
```

To target a specific version of .NET Framework, assign one of the following string values to the `version` attribute:

.NET Framework version	version string
4.8 (including 4.8.1)	v4.0
4.7 (including 4.7.1 and 4.7.2)	v4.0
4.6 (including 4.6.1 and 4.6.2)	v4.0
4.5 (including 4.5.1 and 4.5.2)	v4.0
4	v4.0
3.5	v2.0.50727
2.0	v2.0.50727
1.1	v1.1.4322
1.0	v1.0.3705

Obsolete APIs for .NET Framework 4.5 and later versions

- [Obsolete Types](#)
- [Obsolete Members](#)

Obsolete APIs for previous versions

- [Obsolete Types in .NET Framework 4](#)
- [Obsolete Members in .NET Framework 4](#)
- [.NET Framework 3.5 Obsolete List](#)
- [.NET Framework 2.0 Obsolete List](#)

See also

- [<supportedRuntime> Element](#)

Obsolete types in .NET Framework

Article • 03/31/2023

The tables in this article list the types that are obsolete in the .NET Framework 4.5 and .NET Framework 4.6, organized by assembly. Use the following links to see a list of the obsolete types and the recommended alternatives in each assembly. Because these types are obsolete, all their members are also obsolete. For a list of additional obsolete members in the .NET Framework class library, see [Obsolete Members](#).

- [Obsolete types in system assemblies](#)
 - [mscorlib.dll](#)
 - [System.Core.dll](#)
 - [System.Data.dll](#)
 - [System.Data.OracleClient.dll](#)
 - [System.Design.dll](#)
 - [System.dll](#)
 - [System.EnterpriseServices.dll](#)
 - [System.Net.dll](#)
 - [System.ServiceModel.dll](#)
 - [System.Web.dll](#)
 - [System.Web.Mobile.dll](#)
 - [System.Workflow.Activities.dll](#)
 - [System.Workflow.ComponentModel.dll](#)
 - [System.Workflow.Runtime.dll](#)
 - [System.WorkflowServices.dll](#)
 - [System.Xaml.dll](#)
 - [System.Xml.dll](#)
 - [WindowsBase.dll](#)
- [Obsolete types in Microsoft assemblies](#)
 - [IEHost.dll](#) and [IEExec.exe](#)
 - [Microsoft.Build.Engine.dll](#)
 - [Microsoft.JScript.dll](#)
 - [Microsoft.VisualBasic.Compatibility.dll](#)
 - [Microsoft.VisualBasic.Compatibility.Data.dll](#)
 - [Microsoft.VisualC.dll](#)

Obsolete types in system assemblies

The following tables list the types that have been declared obsolete in system assemblies. These assemblies are used for general-purpose application development that targets .NET Framework.

Assembly: mscorlib.dll

Type	Message
System.ExecutionEngineException	This type previously indicated an unspecified fatal error in the runtime. The runtime no longer raises this exception so this type is obsolete.
System.Collections.CaseInsensitiveHashCodeProvider	Please use System.StringComparer instead.
System.Collections.IHashCodeProvider	Please use System.Collections.IEqualityComparer instead.
System.Configuration.Assemblies.AssemblyHash	The AssemblyHash class has been deprecated.
System.Diagnostics.Contracts.Internal.ContractHelper	First deprecated in .NET Framework 4.5. Use the System.Runtime.CompilerServices.ContractHelper class in the System.Runtime.CompilerServices namespace instead.

Type	Message
System.Reflection.Emit.UnmanagedMarshal	An alternate API is available: Emit the System.Runtime.InteropServices.MarshalAsAttribute custom attribute instead.
System.Runtime.InteropServices.BIND_OPTS	Use System.Runtime.InteropServices.ComTypes.BIND_OPTS instead.
System.Runtime.InteropServices.BINDPTR	Use System.Runtime.InteropServices.ComTypes.BINDPTR instead.
System.Runtime.InteropServices.CALLCONV	Use System.Runtime.InteropServices.ComTypes.CALLCONV instead.
System.Runtime.InteropServices.CONNECTDATA	Use System.Runtime.InteropServices.ComTypes.CONNECTDATA instead.
System.Runtime.InteropServices.DESCKIND	Use System.Runtime.InteropServices.ComTypes.DESCKIND instead.
System.Runtime.InteropServices.DISPPARAMS	Use System.Runtime.InteropServices.ComTypes.DISPPARAMS instead.
System.Runtime.InteropServices.ELEMDESC	Use System.Runtime.InteropServices.ComTypes.ELEMDESC instead.
System.Runtime.InteropServices.EXCEPINFO	Use System.Runtime.InteropServices.ComTypes.EXCEPINFO instead.
System.Runtime.InteropServices.FILETIME	Use System.Runtime.InteropServices.ComTypes.FILETIME instead.
System.Runtime.InteropServices.FUNCDESC	Use System.Runtime.InteropServices.ComTypes.FUNCDESC instead.
System.Runtime.InteropServices.FUNCFLAGS	Use System.Runtime.InteropServices.ComTypes.FUNCFLAGS instead.
System.Runtime.InteropServices.FUNCKIND	Use System.Runtime.InteropServices.ComTypes.FUNCKIND instead.
System.Runtime.InteropServices.IDispatchImplAttribute	This attribute is deprecated and will be removed in a future version.
System.Runtime.InteropServices.IDispatchImplType	The System.Runtime.InteropServices.IDispatchImplAttribute is deprecated.
System.Runtime.InteropServices.IDLDESC	Use System.Runtime.InteropServices.ComTypes.IDLDESC instead.
System.Runtime.InteropServices.IDLFLAG	Use System.Runtime.InteropServices.ComTypes.IDLFLAG instead.
System.Runtime.InteropServices.IMPLTYPEFLAGS	Use System.Runtime.InteropServices.ComTypes.IMPLTYPEFLAGS instead.
System.Runtime.InteropServices.INVOKEKIND	Use System.Runtime.InteropServices.ComTypes.INVOKEKIND instead.
System.Runtime.InteropServices.LIBFLAGS	Use System.Runtime.InteropServices.ComTypes.LIBFLAGS instead.
System.Runtime.InteropServices.PARAMDESC	Use System.Runtime.InteropServices.ComTypes.PARAMDESC instead.
System.Runtime.InteropServices.PARAMFLAG	Use System.Runtime.InteropServices.ComTypes.PARAMFLAG instead.
System.Runtime.InteropServices.SetWin32ContextInIDispatchAttribute	This attribute has been deprecated. Application Domains no longer respect Activation Context boundaries in IDispatch calls.
System.Runtime.InteropServices.STATSTG	Use System.Runtime.InteropServices.ComTypes.STATSTG instead.
System.Runtime.InteropServices.SYSKIND	Use System.Runtime.InteropServices.ComTypes.SYSKIND instead.
System.Runtime.InteropServices.TYPEATTR	Use System.Runtime.InteropServices.ComTypes.TYPEATTR instead.
System.Runtime.InteropServices.TYPEDESC	Use System.Runtime.InteropServices.ComTypes.TYPEDESC instead.
System.Runtime.InteropServices.TYPEFLAGS	Use System.Runtime.InteropServices.ComTypes.TYPEFLAGS instead.
System.Runtime.InteropServices.TYPEKIND	Use System.Runtime.InteropServices.ComTypes.TYPEKIND instead.
System.Runtime.InteropServices.TYPERIBATTR	Use System.Runtime.InteropServices.ComTypes.TYPERIBATTR instead.
System.Runtime.InteropServices.UCOMIBindCtx	Use System.Runtime.InteropServices.ComTypes.IBindCtx instead.
System.Runtime.InteropServices.UCOMIConnectionPoint	Use System.Runtime.InteropServices.ComTypes.IConnectionPoint instead.
System.Runtime.InteropServices.UCOMIConnectionPointContainer	Use System.Runtime.InteropServices.ComTypes.IConnectionPointContainer instead.
System.Runtime.InteropServices.UCOMIEnumConnectionPoints	Use System.Runtime.InteropServices.ComTypes.IEnumConnectionPoints instead.

Type	Message
System.Runtime.InteropServices.UCOMIEnumConnections	Use System.Runtime.InteropServices.ComTypes.IEnumConnections instead.
System.Runtime.InteropServices.UCOMIEnumMoniker	Use System.Runtime.InteropServices.ComTypes.IEnumMoniker instead.
System.Runtime.InteropServices.UCOMIEnumString	Use System.Runtime.InteropServices.ComTypes.IEnumString instead.
System.Runtime.InteropServices.UCOMIEnumVARIANT	Use System.Runtime.InteropServices.ComTypes.IEnumVARIANT instead.
System.Runtime.InteropServices.UCOMIMoniker	Use System.Runtime.InteropServices.ComTypes.IMoniker instead.
System.Runtime.InteropServices.UCOMIPersistFile	Use System.Runtime.InteropServices.ComTypes.IPersistFile instead.
System.Runtime.InteropServices.UCOMIRunningObjectTable	Use System.Runtime.InteropServices.ComTypes.IRunningObjectTable instead.
System.Runtime.InteropServices.UCOMIStream	Use System.Runtime.InteropServices.ComTypes.IStream instead.
System.Runtime.InteropServices.UCOMITypeComp	Use System.Runtime.InteropServices.ComTypes.ITypeComp instead.
System.Runtime.InteropServices.UCOMITypeInfo	Use System.Runtime.InteropServices.ComTypes.ITypeInfo instead.
System.Runtime.InteropServices.UCOMITypeLib	Use System.Runtime.InteropServices.ComTypes.ITypeLib instead.
System.Runtime.InteropServices.VARDESC	Use System.Runtime.InteropServices.ComTypes.VARDESC instead.
System.Runtime.InteropServices.VARFLAGS	Use System.Runtime.InteropServices.ComTypes.VARFLAGS instead.
System.Security.SecurityCriticalScope	SecurityCriticalScope is only used for .NET Framework 2.0 transparency compatibility.
System.Security.SecurityTreatAsSafeAttribute	SecurityTreatAsSafeAttribute is only used for .NET Framework 2.0 transparency compatibility. Please use the System.Security.SecuritySafeCriticalAttribute instead.
System.Security.Policy.FirstMatchCodeGroup	This type is obsolete and will be removed in a future release of the .NET Framework.
System.Security.Policy.PermissionRequestEvidence	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.Security.Policy.UnionCodeGroup	This type is obsolete and will be removed in a future release of the .NET Framework.

[Back to top](#)

Assembly: System.Core.dll

Type	Message
System.Runtime.CompilerServices.ExecutionScope	<p>Use of this type generates a compiler error.</p> <p>Do not use this type.</p>

[Back to top](#)

Assembly: System.Data.dll

Type	Message
System.Data.DataSysDescriptionAttribute	DataSysDescriptionAttribute has been deprecated.
System.Data.PropertyAttributes	PropertyAttributes has been deprecated.
System.Data.TypedDataSetGenerator	The TypedDataSetGenerator class will be removed in a future release. Please use System.Data.Design.TypedDataSetGenerator in System.Design.dll .
System.Xml.XmlDataDocument	The XmlDataDocument class will be removed in a future release.

[Back to top](#)

Assembly: System.Data.OracleClient.dll

Type	Message
System.Data.OracleClient.OracleClientFactory	OracleClientFactory has been deprecated.
System.Data.OracleClient.OracleCommand	OracleCommand has been deprecated.
System.Data.OracleClient.OracleCommandBuilder	OracleCommandBuilder has been deprecated.
System.Data.OracleClient.OracleConnection	OracleConnection has been deprecated.
System.Data.OracleClient.OracleConnectionStringBuilder	OracleConnectionStringBuilder has been deprecated.
System.Data.OracleClient.OracleDataAdapter	OracleDataAdapter has been deprecated.
System.Data.OracleClient.OraclePermission	OraclePermission has been deprecated.
System.Data.OracleClient.OraclePermissionAttribute	System.Data.OracleClient.OraclePermissionAttribute has been deprecated.

[Back to top](#)

Assembly: System.Design.dll

Type	Message
System.ComponentModel.Design.LocalizationExtenderProvider	This class has been deprecated. Use System.ComponentModel.Design.Serialization.CodeDomLocalizationProvider instead.
System.Web.UI.Design.DataBindCollectionConverter	Use of this type is not recommended because DataBindings editing is launched via a System.ComponentModel.Design.DesignerActionList instead of the property grid.
System.Web.UI.Design.DataBindCollectionEditor	Use of this type is not recommended because DataBindings editing is launched via a System.ComponentModel.Design.DesignerActionList instead of the property grid.
System.Web.UI.Design.IControlDesignerBehavior	The recommended alternative is System.Web.UI.Design.IControlDesignerTag and System.Web.UI.Design.IControlDesignerView .
System.Web.UI.Design.IHtmlControlDesignerBehavior	The recommended alternative is System.Web.UI.Design.IControlDesignerTag and System.Web.UI.Design.IControlDesignerView .
System.Web.UI.Design.ITemplateEditingFrame	Use of this type is not recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags .
System.Web.UI.Design.IWebFormReferenceManager	The recommended alternative is System.Web.UI.Design.WebFormsReferenceManager . The WebFormsReferenceManager contains additional functionality and allows for more extensibility. To get the WebFormsReferenceManager , use the RootDesigner.ReferenceManager property from your System.Web.UI.Design.ControlDesigner .
System.Web.UI.Design.IWebFormsDocumentService	The recommended alternative is System.Web.UI.Design.WebFormsRootDesigner . The WebFormsRootDesigner contains additional functionality and allows for more extensibility. To get the WebFormsRootDesigner , use the RootDesigner property from your System.Web.UI.Design.ControlDesigner .
System.Web.UI.Design.ITemplateEditingService	Use of this type is not recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags .

Type	Message
System.Web.UI.Design.ReadWriteControlDesigner	The recommended alternative is System.Web.UI.Design.ContainerControlDesigner because it uses an System.Web.UI.Design.EditableDesignerRegion for editing the content. Designer regions allow for better control of the content being edited.
System.Web.UI.Design.TemplateEditingService	Use of this type is not recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags .
System.Web.UI.Design.TemplateEditingVerb	Use of this type is not recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags .
System.Web.UI.Design.WebControls.CalendarAutoFormatDialog	Use of this type is not recommended because the AutoFormat dialog is launched by the designer host. The list of available AutoFormats is exposed on the System.Web.UI.Design.ControlDesigner in the ControlDesigner.AutoFormats property.
System.Web.UI.Design.WebControls.PanelDesigner	The recommended alternative is System.Web.UI.Design.WebControls.PanelContainerDesigner because it uses an System.Web.UI.Design.EditableDesignerRegion for editing the content. Designer regions allow for better control of the content being edited.

[Back to top](#)

Assembly: System.dll

Type	Message
System.ComponentModel.IComNativeDescriptorHandler	This interface has been deprecated. Add a System.ComponentModel.TypeDescriptionProvider to handle type TypeDescriptor.ComObjectType instead.
System.ComponentModel.RecommendedAsConfigurableAttribute	Use System.ComponentModel.SettingsBindableAttribute instead to work with the new settings model.
System.ComponentModel.Design.Serialization.RootDesignerSerializerAttribute	This attribute has been deprecated. Use System.ComponentModel.Design.Serialization.DesignerSerializerAttribute instead.
System.Diagnostics.DiagnosticsConfigurationHandler	This class has been deprecated.
System.Diagnostics.PerformanceCounterManager	This class has been deprecated. Use the performance counters through the System.Diagnostics.PerformanceCounter class instead.
System.Net.GlobalProxySelection	This class has been deprecated. Please use WebRequest.DefaultWebProxy instead to access and set the global default proxy. Use 'null' instead of GlobalProxySelection.GetEmptyWebProxy .
System.Net.Sockets.SocketClientAccessPolicyProtocol	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.

[Back to top](#)

Assembly: System.EnterpriseServices.dll

Type	Message
System.EnterpriseServices.RegistrationHelperTx	The RegistrationHelperTx class has been deprecated.

[Back to top](#)

Assembly: System.Net.dll

Type	Message
System.Net.INetworkProgress	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.IUnsafeWebRequestCreate	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.NetworkProgressChangedEventArgs	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.UiSynchronizationContext	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.Sockets.HttpPolicyDownloaderProtocol	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.Sockets.SecurityCriticalAction	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.Sockets.SocketPolicy	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.Sockets.UdpAnySourceMulticastClient	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Net.Sockets.UdpSingleSourceMulticastClient	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.

[Back to top](#)

Assembly: System.ServiceModel.dll

Type	Message
System.ServiceModel.NetPeerTcpBinding	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Channels.HttpCookieContainerBindingElement	First deprecated in the .NET Framework 4.5. This type is obsolete. To enable Http CookieContainer , use the <code>AllowCookies</code> property on the Http binding or on the HttpTransportBindingElement .
System.ServiceModel.Channels.PeerCustomResolverBindingElement	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Channels.PeerTransportBindingElement	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Configuration.NetPeerTcpBindingCollectionElement	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Configuration.NetPeerTcpBindingElement	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Configuration.PeerTransportElement	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.PeerResolvers.CustomPeerResolverService	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.

[Back to top](#)

Assembly: System.Web.dll

Type	Message
System.Web.Configuration.PassportAuthentication	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account ↗
System.Web.Mail.MailAttachment	The recommended alternative is System.Net.Mail.Attachment .
System.Web.Mail.MailEncoding	The recommended alternative is System.Net.Mime.TransferEncoding .
System.Web.Mail.MailFormat	The recommended alternative is MailMessage.IsBodyHtml .
System.Web.Mail.MailMessage	The recommended alternative is System.Net.Mail.MailMessage .
System.Web.Mail.MailPriority	The recommended alternative is System.Net.Mail.MailPriority .
System.Web.Mail.SmtpMail	The recommended alternative is System.Net.Mail.SmtpClient .
System.Web.Security.PassportAuthenticationEventArgs	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account ↗

Type	Message
System.Web.Security.PassportAuthenticationEventHandler	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.Security.PassportAuthenticationModule	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.Security.PassportIdentity	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.Security.PassportPrincipal	This type is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.UI.ObjectConverter	The recommended alternative is System.Convert and String.Format .

[Back to top](#)

Assembly: System.Web.Mobile.dll

Type	Message
System.Web.Mobile.CookielessData	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.DeviceFilterElement	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.DeviceFilterElementCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.DeviceFiltersSection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.ErrorHandlerModule	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.MobileCapabilities	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.MobileDeviceCapabilitiesSectionHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.MobileErrorInfo	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.Mobile.MobileFormsAuthentication	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.Design.MobileControls.IMobileDesigner	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.Design.MobileControls.IMobileWebFormServices	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.Design.MobileControls.MobileResource	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.Design.MobileControls.Converters.DataFieldConverter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.Design.MobileControls.Converters.DataMemberConverter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.AdRotator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Alignment	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ArrayListCollectionBase	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.BaseValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.BooleanOption	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Calendar	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Command	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.CommandFormat	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.CompareValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Constants	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ControlElement	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ControlElementCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ControlPager	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.CustomValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DesignerAdapterAttribute	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceElement	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceElementCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceOverridableAttribute	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecific	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecificChoice	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecificChoiceCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecificChoiceControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecificChoiceTemplateBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.DeviceSpecificChoiceTemplateContainer	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.DeviceSpecificControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ErrorFormatterPage	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.FontInfo	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.FontSize	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Form	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.FormControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.FormMethod	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.IControlAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Image	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.IObjectListFieldCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.IPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ItemPager	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ITemplateable	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Label	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Link	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.List	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListCommandEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListCommandEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListDataBindEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListDataBindEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListDecoration	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ListSelectType	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.LiteralLink	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.LiteralText	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.LiteralTextContainerControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.LiteralTextControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.LoadItemsEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.LoadItemsEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileControl	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileControlsSection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileControlsSectionHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileListItem	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileListItemCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileListItemType	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobilePage	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileTypeNameConverter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.MobileUserControl	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectList	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListCommand	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.ObjectListCommandCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListCommandEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListCommandEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListDataBindEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListDataBindEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListField	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListFieldCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListFormItem	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListFormItemCollection	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListSelectEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListSelectEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListShowCommandsEventArgs	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListShowCommandsEventHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.ObjectListTitleAttribute	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ObjectListViewMode	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.PagedControl	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.PagerStyle	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Panel	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.PanelControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.PersistNameAttribute	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.PhoneCall	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.RangeValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.RegularExpressionValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.RequiredFieldValidator	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.SelectionList	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Style	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.StyleSheet	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.StyleSheetControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TemplateContainer	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TextBox	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TextBoxControlBuilder	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TextControl	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TextView	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.TextViewElement	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.ValidationSummary	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Wrapping	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlCalendarAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlCommandAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlFormAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlImageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlLinkAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.ChtmlMobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlPhoneCallAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlSelectionListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ChtmlTextBoxAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.ControlAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlCalendarAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlCommandAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlControlAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlFormAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlImageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlLabelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlLinkAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.HtmlLiteralTextAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlMobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlObjectListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlPanelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlPhoneCallAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlSelectionListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlTextBoxAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlTextListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlValidationSummaryAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.HtmlValidatorAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.MobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.MultiPartWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.UpWmlMobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.UpWmlPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlCalendarAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlCommandAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlControlAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlFormAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlImageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlLabelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlLinkAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlLiteralTextAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlMobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlObjectListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlPanelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.WmlPhoneCallAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlPostFieldType	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlSelectionListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlTextBoxAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlTextListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlValidationSummaryAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.WmlValidatorAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.Doctype	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.StyleSheetLocation	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlCalendarAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlCommandAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlControlAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlCssHandler	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlFormAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlImageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlLabelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlLinkAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlLiteralTextAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlMobileTextWriter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlObjectListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlPageAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlPanelAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlPhoneCallAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlSelectionListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlTextBoxAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlTextListAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlValidationSummaryAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

Type	Message
System.Web.UI.MobileControls.Adapters.XhtmlAdapters.XhtmlValidatorAdapter	The System.Web.Mobile.dll assembly has been deprecated and should no longer be used. For information about how to develop ASP.NET mobile applications, see ASP.NET for Mobiles .

[Back to top](#)

Assembly: System.Workflow.Activities.dll

Type	Message
All types in the System.Workflow.Activities namespace	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.Activities.Configuration.ActiveDirectoryRoleFactoryConfiguration	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.Activities.Rules.RuleActionTrackingEvent	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.Activities.Rules.RuleConditionReference	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.Activities.Rules.RuleSetReference	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .

[Back to top](#)

Assembly: System.Workflow.ComponentModel.dll

Type	Message
All types in the System.Workflow.ComponentModel namespace except System.Workflow.ComponentModel.GetValueOverride and System.Workflow.ComponentModel.SetValueOverride	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
All types in the System.Workflow.ComponentModel.Compiler namespace except System.Workflow.ComponentModel.Compiler.ValidationError and System.Workflow.ComponentModel.Compiler.ValidationErrorCollection	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
All types in the System.Workflow.ComponentModel.Design namespace except ConnectorEventHandler	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.ActivityCodeDomSerializationManager	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .

Type	Message
System.Workflow.ComponentModel.Serialization.ActivityCodeDomSerializer	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.ActivityMarkupSerializer	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.ActivitySurrogateSelector	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.ActivityTypeCodeDomSerializer	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.CompositeActivityMarkupSerializer	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
System.Workflow.ComponentModel.Serialization.DependencyObjectCodeDomSerializer	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .

[Back to top](#)

Assembly: System.Workflow.Runtime.dll

Type	Message
System.Activities.Statements.Interop	First deprecated in the .NET Framework 4.5. The Workflow Foundation 3.0 types are deprecated. Instead, use the Workflow 4.0 types from System.Activities.* .
System.Activities.Tracking.InteropTrackingRecord	First deprecated in the .NET Framework 4.5. The Workflow Foundation 3.0 types are deprecated. Instead, use the Workflow 4.0 types from System.Activities.* .
All types in the System.Workflow.Runtime namespace	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
All types in the System.Workflow.Runtime.Configuration namespace	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .
All types in the System.Workflow.Runtime.DebugEngine namespace except DebugEngineCallback	First deprecated in the .NET Framework 4.5. The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.* .

Type	Message
All types in the System.Workflow.Runtime.Hosting namespace except WorkflowCommitWorkBatchService.CommitWorkBatchCallback	<p>First deprecated in the .NET Framework 4.5.</p> <p>The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.*.</p>
All types in the System.Workflow.Runtime.Tracking namespace	<p>First deprecated in the .NET Framework 4.5.</p> <p>The System.Workflow.* types are deprecated. Instead, please use the new types from System.Activities.*.</p>

[Back to top](#)

Assembly: System.WorkflowServices.dll

Type	Message
System.ServiceModel.WorkflowServiceHost	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Activation.WorkflowServiceHostFactory	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Activities.Description.WorkflowRuntimeEndpoint	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Configuration.ExtendedWorkflowRuntimeServiceElementCollection	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Configuration.PersistenceProviderElement	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Configuration.WorkflowRuntimeElement	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Description.DurableOperationAttribute	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Description.DurableServiceAttribute	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Description.PersistenceProviderBehavior	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>
System.ServiceModel.Description.UnknownExceptionAction	<p>First deprecated in the .NET Framework 4.5.</p> <p>The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.*.</p>

Type	Message
System.ServiceModel.Description.WorkflowRuntimeBehavior	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Dispatcher.DurableOperationContext	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.InstanceLockException	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.InstanceNotFoundException	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.LockingPersistenceProvider	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.PersistenceException	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.PersistenceProvider	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.PersistenceProviderFactory	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.ServiceModel.Persistence.SqlPersistenceProviderFactory	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
All types in the System.Workflow.Activities namespace	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .
System.Workflow.Runtime.Hosting.ChannelManagerService	First deprecated in the .NET Framework 4.5. The WF 3 types are deprecated. Instead, please use the new WF 4 types from System.Activities.* .

[Back to top](#)

Assembly: System.Xaml.dll

Type	Message
System.Windows.Markup.AcceptedMarkupExtensionExpressionTypeAttribute	This is not used by the XAML parser. Please look at System.Windows.Markup.XamlSetMarkupExtensionAttribute .

[Back to top](#)

Assembly: System.Xml.dll

Type	Message
System.Xml.IApplicationResourceStreamResolver	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Xml.Schema.XmlSchemaCollection	Use System.Xml.Schema.XmlSchemaSet for schema compilation and validation.
System.Xml.XmlValidatingReader	Use an System.Xml.XmlReader created by the XmlReader.Create method using the appropriate System.Xml.XmlReaderSettings instead.
System.Xml.XmlXapResolver	Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is not intended to be used directly from your code.
System.Xml.Xsl.XsltTransform	This class has been deprecated. Please use System.Xml.Xsl.XslCompiledTransform instead.

[Back to top](#)

Assembly: WindowsBase.dll

Type	Message
System.Windows.Markup.IReceiveMarkupExtension	System.Windows.Markup.IReceiveMarkupExtension has been deprecated. This interface is no longer in use.

[Back to top](#)

Obsolete types in Microsoft assemblies

The following sections list the obsolete types in Microsoft assemblies. These assemblies are special-purpose assemblies, such as assemblies that target an individual language (for example, Microsoft.JScript.dll or Microsoft.VisualC.dll).

Assembly: IEHost.dll and IEEExec.exe

The IEHost.dll and IEEExec.exe assemblies have been removed from .NET Framework. All of their types and their members are obsolete and are not supported as of .NET Framework 4. These assemblies were used to host Windows Forms controls and to run executables in Internet Explorer. Recommended alternatives include ClickOnce, XAML browser applications (XBAP), and Microsoft Silverlight.

[Back to top](#)

Assembly: Microsoft.Build.Engine.dll

Type	Message
Microsoft.Build.BuildEngine.Engine	This class has been deprecated. Please use Microsoft.Build.Evaluation.ProjectCollection from the <i>Microsoft.Build</i> assembly instead.
Microsoft.Build.BuildEngine.Project	This class has been deprecated. Please use Microsoft.Build.Evaluation.ProjectCollection from the <i>Microsoft.Build</i> assembly instead.

[Back to top](#)

Assembly: Microsoft.JScript.dll

Type	Message
Microsoft.JScript.Vsa.BaseVsaEngine	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.

Type	Message
Microsoft.JScript.Vsa.BaseVsaSite	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.BaseVsaStartup	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaCodeItem	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaEngine	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaError	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaGlobalItem	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaItem	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaItems	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaPersistSite	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaReferenceItem	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.IJSVsaSite	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.JSVsaError	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.JSVsaException	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.JSVsaItemFlag	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.JSVsaItemType	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.ResInfo	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.
Microsoft.JScript.Vsa.VsaEngine	This type was deprecated in Visual Studio 2005; there is no replacement for this feature. Please see the System.CodeDom.Compiler.ICodeCompiler documentation for additional help.

[Back to top](#)

Assembly: Microsoft.VisualBasic.Compatibility.dll

For information about migrating from Visual Basic 6, see [Visual Basic 6.0 Resource Center](#).

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.BaseControlArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.BaseOcxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ButtonArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.CheckBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.CheckedListBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ColorDialogArray	This member is obsolete.

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.ComboBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DirListBox	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DirListBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DriveListBox	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DriveListBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.FileListBox	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.FileListBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.FixedLengthString	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.FontDialogArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.FormShowConstants	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.GroupBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.HScrollBarArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ImageListArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.LabelArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ListBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ListBoxItem	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ListViewArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.LoadResConstants	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.MaskedTextBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.MenuItemArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.MouseButtonConstants	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.OpenFileDialogArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.PanelArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.PictureBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.PrintDialogArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ProgressBarArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.RadioButtonArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.RichTextBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.SaveFileDialogArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ScaleMode	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ShiftConstants	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.StatusBarArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.StatusStripArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.Support	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.TabControlArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.TextBoxArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.TimerArray	This member is obsolete.

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.ToolBarArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ToolStripArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ToolStripItemArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.TreeViewArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.VScrollBarArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebBrowserArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClass	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassContainingClassNotOptional	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassCouldNotFindEvent	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassNextItemCannotBeCurrentWebItem	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassNextItemRespondNotFound	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassUserWebClassNameNotOptional	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassWebClassNameNotOptional	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebClassWebItemNotValid	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItem	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemAssociatedWebClassNotOptional	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemClosingTagNotFound	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemCouldNotLoadEmbeddedResource	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemCouldNotLoadTemplateFile	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemNameNotOptional	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemNoTemplateSpecified	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemTooManyNestedTags	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.WebItemUnexpectedErrorReadingTemplateFile	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ZOrderConstants	This member is obsolete.

[Back to top](#)

Assembly: Microsoft.VisualBasic.Compatibility.Data.dll

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.ADODC	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.BOFActionEnum	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.EndOfRecordsetDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.EOFActionEnum	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.ErrorDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.FetchCompleteDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.FetchProgressDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.FieldChangeCompleteDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.MoveCompleteDelegate	This member is obsolete.

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.ADODC.OrientationEnum	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.RecordChangeCompleteDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.RecordsetChangeCompleteDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.WillChangeFieldDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.WillChangeRecordDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.WillChangeRecordsetDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODC.WillMoveDelegate	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.ADODCArray	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.BaseDataEnvironment	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.BindingCollectionEnumerator	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.CONNECTDATA	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBBINDING	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBCOLUMNINFO	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBID	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBBinding	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBBindingCollection	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBKINDENUM	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.DBPROPIDSET	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IAccessor	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IChaperonedRowset	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IColorsInfo	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IConnectionPoint	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IConnectionPointContainer	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IDataFormat	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IDataFormatDisp	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IEnumConnectionPoints	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IEnumConnections	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowPosition	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowPositionChange	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowset	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowsetChange	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowsetIdentity	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowsetInfo	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.IRowsetNotify	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.MBinding	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.MBindingCollection	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.SRDescriptionAttribute	This member is obsolete.

Type	Message
Microsoft.VisualBasic.Compatibility.VB6.UGUID	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.UNAME	This member is obsolete.
Microsoft.VisualBasic.Compatibility.VB6.UpdateMode	This member is obsolete.

[Back to top](#)

Assembly: Microsoft.VisualBasic.dll

Type	Message
Microsoft.VisualBasic.DebugInfoInPDBAttribute	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.DecoratedNameAttribute	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.IsConstModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.IsCXXReferenceModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.IsLongModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.IsSignedModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.IsVolatileModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.MiscellaneousBitsAttribute	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.NeedsCopyConstructorModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.
Microsoft.VisualBasic.NoSignSpecifiedModifier	Microsoft.VisualBasic.dll is an obsolete assembly and exists only for backwards compatibility.

See also

- [What's Obsolete in the Class Library](#)
- [Obsolete Members](#)

Obsolete members in .NET Framework

Article • 03/31/2023

This article lists the type members that are obsolete in .NET Framework 4.5 and later versions. Type members are grouped by assembly.

This article doesn't list the members of obsolete types. For a list of obsolete types, see [Obsolete types](#).

mscorlib.dll

Type	Member	Message
Microsoft.Win32.Registry	DynData	The DynData registry key only works on Win9x, which is no longer supported by the CLR. On NT-based operating systems, use the Registry.PerformanceData registry key or the RegistryProxy.PerformanceData registry proxy instead.
System.Activator	CreateInstance(AppDomain, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of Activator.CreateInstance that doesn't take an System.Security.Policy.Evidence parameter.
System.Activator	CreateInstance(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of Activator.CreateInstance that doesn't take an System.Security.Policy.Evidence parameter.
System.Activator	CreateInstanceFrom(AppDomain, String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use Evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of Activator.CreateInstanceFrom that doesn't take an System.Security.Policy.Evidence parameter.
System.Activator	CreateInstanceFrom(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of Activator.CreateInstanceFrom that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	AppendPrivatePath	AppendPrivatePath has been deprecated. Investigate the use of AppDomainSetup.PrivateBinPath instead.

Type	Member	Message
System.AppDomain	ClearPrivatePath	ClearPrivatePath has been deprecated. Investigate the use of AppDomainSetup.PrivateBinPath instead.
System.AppDomain	ClearShadowCopyPath	ClearShadowCopyPath has been deprecated. Investigate the use of AppDomainSetup.ShadowCopyDirectories instead.
System.AppDomain	CreateInstance(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.CreateInstance that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	CreateInstanceAndUnwrap(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.CreateInstanceAndUnwrap that doesn't take a System.Security.Policy.Evidence parameter.
System.AppDomain	CreateInstanceFrom(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.CreateInstanceFrom that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	CreateInstanceFromAndUnwrap(String, String, Boolean, BindingFlags, Binder, Object[], CultureInfo, Object[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.CreateInstanceFromAndUnwrap that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, Evidence)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, Evidence, PermissionSet, PermissionSet, PermissionSet, PermissionSet)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, PermissionSet, PermissionSet, PermissionSet)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.

Type	Member	Message
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, String, Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.DefineDynamicAssembly that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, String, Evidence, PermissionSet, PermissionSet, PermissionSet)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, String, Evidence, PermissionSet, PermissionSet, PermissionSet, PermissionSet, Boolean)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, String, Evidence, PermissionSet, PermissionSet, PermissionSet, PermissionSet, Boolean, IEnumerable<CustomAttributeBuilder>)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess, String, PermissionSet, PermissionSet, PermissionSet)	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.AppDomain	ExecuteAssembly(String, Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssembly that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	ExecuteAssembly(String, Evidence, String[])	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssembly that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	ExecuteAssembly(String, Evidence, String[], Byte[], AssemblyHashAlgorithm)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssembly that doesn't take an System.Security.Policy.Evidence parameter.

Type	Member	Message
System.AppDomain	ExecuteAssemblyByName(AssemblyName, Evidence, String[])	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssemblyByName that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	ExecuteAssemblyByName(String, Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssemblyByName that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	ExecuteAssemblyByName(String, Evidence, String[])	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.ExecuteAssemblyByName that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	GetCurrentThreadId	GetCurrentThreadId has been deprecated because it doesn't provide a stable ID when managed threads are running on fibers (also known as light weight threads). To get a stable identifier for a managed thread, use the Thread.ManagedThreadId property.
System.AppDomain	Load(AssemblyName, Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	Load(Byte[], Byte[], Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	Load(String, Evidence)	Methods that use evidence to sandbox are obsolete and will be removed in a future release of the .NET Framework. Use an overload of AppDomain.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.AppDomain	SetAppDomainPolicy	AppDomain policy levels are obsolete and will be removed in a future release of the .NET Framework.

Type	Member	Message
System.AppDomain	SetCachePath	<code>SetCachePath</code> has been deprecated. Investigate the use of <code>AppDomainSetup.CachePath</code> instead.
System.AppDomain	SetDynamicBase	<code>SetDynamicBase</code> has been deprecated. Investigate the use of <code>AppDomainSetup.DynamicBase</code> instead.
System.AppDomain	SetShadowCopyFiles	<code>SetShadowCopyFiles</code> has been deprecated. Investigate the use of <code>AppDomainSetup.ShadowCopyFiles</code> instead.
System.AppDomain	SetShadowCopyPath	<code>SetShadowCopyPath</code> has been deprecated. Investigate the use of <code>AppDomainSetup.ShadowCopyDirectories</code> instead.
System.Enum	ToString(IFormatProvider)	The <code>provider</code> argument isn't used. Use <code>Enum.ToString()</code> .
System.Enum	ToString(String, IFormatProvider)	The <code>provider</code> argument isn't used. Use <code>Enum.ToString(String)</code> .
System.LoaderOptimization	DisallowBindings	This method has been deprecated. Use <code>Assembly.Load</code> instead.
System.LoaderOptimization	DomainMask	This method has been deprecated. Use <code>Assembly.Load</code> instead.
System.Collections.Hashtable	Hashtable(IDictionary, IHashCodeProvider, IComparer)	Use <code>Hashtable(IDictionary, IEqualityComparer)</code> instead.
System.Collections.Hashtable	Hashtable(IHashCodeProvider, IComparer)	Use <code>Hashtable(IEqualityComparer)</code> instead.
System.Collections.Hashtable	Hashtable(Int32, IHashCodeProvider, IComparer)	Use <code>Hashtable(Int32, IEqualityComparer)</code> instead.
System.Collections.Hashtable	Hashtable(Int32, Single, IHashCodeProvider, IComparer)	Use <code>Hashtable(Int32, Single, IEqualityComparer)</code> instead.
System.Collections.Hashtable	Hashtable.comparer	Use the <code>Hashtable.EqualityComparer</code> property.

Type	Member	Message
System.Collections.Hashtable	Hashtable.hcp	Use <code>KeyComparer</code> properties.
System.Collections.Hashtable	Hashtable(IDictionary, Single, IHashCodeProvider, IComparer)	Use <code>Hashtable(IDictionary, Single, IEqualityComparer)</code> instead.
System.Configuration.Assemblies.AssemblyHash	Algorithm	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.
System.Configuration.Assemblies.AssemblyHash	AssemblyHash(AssemblyHashAlgorithm, Byte[])	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.
System.Configuration.Assemblies.AssemblyHash	AssemblyHash(Byte[])	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.
System.Configuration.Assemblies.AssemblyHash	Clone	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.
System.Configuration.Assemblies.AssemblyHash	Empty	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.
System.Configuration.Assemblies.AssemblyHash	GetValue	The <code>System.Configuration.Assemblies.AssemblyHash</code> class has been deprecated.

Type	Member	Message
System.Configuration.AssemblyHash	SetValue	The System.Configuration.Assemblies.AssemblyHash class has been deprecated.
System.Diagnostics.Debugger	Debugger	Don't create instances of the Debugger class. Call the static methods directly on this type instead.
System.Diagnostics.StackTrace	StackTrace(Thread, Boolean)	First deprecated in the .NET Framework 4.5.
System.Diagnostics.StackTrace	StackTrac e	This constructor has been deprecated. Use a constructor that doesn't require a Thread parameter.
System.Diagnostics.SymbolStore	ISymbolBinder	The recommended alternative is ISymbolBinder1 . GetReader , which takes the importer interface pointer as an IntPtr instead of an Int32 , and thus works on both 32-bit and 64-bit architectures.
System.Globalization.CultureTypes	FrameworkCultures	This value has been deprecated. Use other values in System.Globalization.CultureTypes .
System.Globalization.CultureTypes	WindowsOnlyCultures	This value has been deprecated. Use other values in System.Globalization.CultureTypes .
System.IO.FileStream	FileStream(IntPtr, FileAccess)	This constructor has been deprecated. Use FileStream(SafeFileHandle, FileAccess) instead.
System.IO.FileStream	FileStream(IntPtr, FileAccess, Boolean)	This constructor has been deprecated. Use FileStream(SafeFileHandle, FileAccess) instead, and optionally make a new Microsoft.Win32.SafeHandles.SafeFileHandle with <code>ownsHandle = false</code> if needed.
System.IO.FileStream	FileStream(IntPtr, FileAccess, Boolean, Int32)	This constructor has been deprecated. Use FileStream(SafeFileHandle, FileAccess, Int32) instead, and optionally make a new Microsoft.Win32.SafeHandles.SafeFileHandle with <code>ownsHandle = false</code> if needed.

Type	Member	Message
System.I O.FileStre am	FileStream(IntPtr, FileAccess, Bool ean, Int32, Boolean)	This constructor has been deprecated. Use FileSt ream(SafeFileHandle, FileAccess, Int32, Boolean) i nstead, and optionally make a new Microsoft.Wi n32.SafeHandles.SafeFileHandle with <code>ownsHandle = false</code> if needed.
System.I O.FileStre am	Handle	This property has been deprecated. Use the FileS tream.SafeFileHandle property instead.
System.I O.Path	InvalidPathChars	Use Path.GetInvalidPathChars or Path.GetInvalidF ileNameChars instead.
System.I O.Stream	CreateWaitHandle	CreateWaitHandle will be removed eventually. Us e <code>new ManualResetEvent(false)</code> instead.
System.I O.Stream	ObjectInvariant	First deprecated in the .NET Framework 4.5. Don't call or override this method.
System.I O.Isolated Storage.Is olatedSto rage	CurrentSize	CurrentSize has been deprecated because it isn't CLS Compliant. To get the current size, use Isolat edStorage.UsedSize .
System.I O.Isolated Storage.Is olatedSto rage	MaximumSize	MaximumSize has been deprecated because it is n't CLS Compliant. To get the maximum size, use IsolatedStorage.Quota .
System.I O.Isolated Storage.Is olatedSto rageFile	CurrentSize	CurrentSize has been deprecated because it isn't CLS Compliant. To get the current size, use Used Size .
System.I O.Isolated Storage.Is olatedSto rageFile	MaximumSize	MaximumSize has been deprecated because it is n't CLS Compliant. To get the maximum size, use IsolatedStorageFile.Quota .

Type	Member	Message
System.IO.IsolatedStorage.IsolatedStorageFileStream	Handle	This property has been deprecated. Use the IsolatedStorageFileStream.SafeFileHandle property instead.
System.Reflection.Assembly	Load(AssemblyName, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	Load(Byte[], Byte[], Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	Load(String, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.Load that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	LoadFile(String, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.LoadFile that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	LoadFrom(String, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.LoadFrom that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	LoadFrom(String, Evidence, Byte[], AssemblyHashAlgorithm)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of Assembly.LoadFrom that doesn't take an System.Security.Policy.Evidence parameter.
System.Reflection.Assembly	LoadWithPartialName(String)	This method has been deprecated. Use Assembly.Load instead.
System.Reflection.Assembly	LoadWithPartialName(String, Evidence)	This method has been deprecated. Use Assembly.Load instead.

Type	Member	Message
System.Reflection.AssemblyFlagsAttribute	AssemblyFlagsAttribute(Int32)	This constructor has been deprecated. Use AssemblyFlagsAttribute(AssemblyNameFlags) instead.
System.Reflection.AssemblyFlagsAttribute	AssemblyFlagsAttribute(UInt32)	This constructor has been deprecated. Use AssemblyFlagsAttribute(AssemblyNameFlags) instead.
System.Reflection.AssemblyFlagsAttribute	Flags	This property has been deprecated. Use AssemblyFlagsAttribute.AssemblyFlags instead.
System.Reflection.Emit.ConstructorBuilder	ReturnType	This property has been deprecated.
System.Reflection.Emit.FieldBuilder	SetMarshal	An alternate API is available: Emit the System.Runtime.InteropServices.MarshalAsAttribute custom attribute instead.
System.Reflection.Emit.FlowControl	Phi	This API has been deprecated.
System.Reflection.Emit.MethodBuilder	SetMarshal	An alternate API is available: Emit the System.Runtime.InteropServices.MarshalAsAttribute custom attribute instead.
System.Reflection.Emit.OpCodeType	Annotation	This API has been deprecated.
System.Reflection.Emit.OperandType	InlinePhi	This API has been deprecated.

Type	Member	Message
System.Reflection.Emit.ParameterBuilder	SetMarshal	An alternate API is available: Emit the System.Runtime.InteropServices.MarshalAsAttribute custom attribute instead.
System.Resources.ResourceManager	ResourceSets	Call ResourceManager.InternalGetResourceSet(CultureInfo, Boolean, Boolean) instead.
System.Runtime.InteropServices.Marshal	GetManagedThunkForUnmanagedMethodPtr	The GetManagedThunkForUnmanagedMethodPtr method has been deprecated and will be removed in a future release.
System.Runtime.InteropServices.Marshal	GetThreadFromFiberCookie	The GetThreadFromFiberCookie method has been deprecated. Use the hosting API to perform this operation.
System.Runtime.InteropServices.Marshal	GetTypeInfoName(UCOMITypeInfo)	Use Marshal.GetTypeInfoName(ITypeInfo) instead.
System.Runtime.InteropServices.Marshal	GetTypeLibGuid(UCOMITypeLib)	Use Marshal.GetTypeLibGuid(ITypeLib) instead.
System.Runtime.InteropServices.Marshal	GetTypeLibLcid(UCOMITypeLib)	Use Marshal.GetTypeLibLcid(ITypeLib) instead.
System.Runtime.InteropServices.Marshal	GetTypeLibName(UCOMITypeLib)	Use Marshal.GetTypeLibName(ITypeLib) instead.

Type	Member	Message
System.Runtime.InteropServices.Marshal	GetUnmanagedThunkForManagedMethodPtr	The GetUnmanagedThunkForManagedMethodPtr method has been deprecated and will be removed in a future release.
System.Runtime.InteropServices.Marshal	ReleaseThreadCache	This API didn't perform any operation and will be removed in future versions of the CLR.
System.Runtime.InteropServices.RuntimeEnvironment		Use of this member generates a compiler error. Don't create instances of the System.Runtime.InteropServices.RuntimeEnvironment class. Call the static methods directly on this type instead.
System.Runtime.Remoting.Channels.ChannelServices	RegisterChannel	Use ChannelServices.RegisterChannel(IChannel, Boolean) instead.
System.Runtime.Remoting.Lifetime.LifetimeServices	LifetimeServices	Use of this member generates a compiler error. Don't create instances of the LifetimeServices class. Call the static methods directly on this type instead.
System.Runtime.Remoting.RemotingConfiguration	Configure(String)	Use RemotingConfiguration.Configure(String, Boolean) instead.
System.Runtime.Remoting.RemotingServices	LogRemotingStage	Use of this method isn't recommended. The LogRemotingStage existed for internal diagnostic purposes only.

Type	Member	Message
System.Security.CodeAccessPermission	Deny	Deny is obsolete and will be removed in a future release of the .NET Framework.
System.Security.CodeAccessPermission	RevertDeny	Deny is obsolete and will be removed in a future release of the .NET Framework.
System.Security.HostSecurityManager	DomainPolicy	AppDomain policy levels are obsolete and will be removed in a future release of the .NET Framework.
System.Security.HostSecurityManagerOptions	HostPolicyLevel	AppDomain policy levels are obsolete and will be removed in a future release of the .NET Framework.
System.Security.PermissionSet	ConvertPermissionSet	This method is obsolete and should no longer be used.
System.Security.PermissionSet	Deny	Deny is obsolete and will be removed in a future release of the .NET Framework.
System.Security.SecurityCriticalAttribute	Scope	System.Security.SecurityCriticalScope is only used for .NET Framework 2.0 transparency compatibility.
System.Security.SecurityManager	CheckExecutionRights	Because execution permission checks can no longer be turned off, the CheckExecutionRights property no longer has any effect.
System.Security.SecurityManager	IsGranted	IsGranted is obsolete and will be removed in a future release of the .NET Framework. Use either the AppDomain.PermissionSet property or the Assembly.PermissionSet property instead.

Type	Member	Message
System.Sec urity.Sec urityMana ger	LoadPolicyLevelFromFile	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	LoadPolicyLevelFromString	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	PolicyHierarchy	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	ResolvePolicy(Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	ResolvePolicy(Evidence, Permissio nSet, PermissionSet, PermissionS et, PermissionSet)	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	ResolvePolicy(Evidence[])	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	ResolvePolicyGroups	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	ResolveSystemPolicy	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	SavePolicy	This method is obsolete and will be removed in a future release of the .NET Framework.
System.Sec urity.Sec urityMana ger	SavePolicyLevel	This method is obsolete and will be removed in a future release of the .NET Framework.

Type	Member	Message
System.Security.Securit yManager	SecurityEnabled	Because security can no longer be turned off, the SecurityEnabled property no longer has any effect.
System.Security.Cryptograph y.PasswordDeriveBy tes	GetBytes	System.Security.Cryptography.Rfc2898DeriveByte s replaces System.Security.Cryptography.PasswordDeriveBytes for deriving key material from a password and is preferred in new applications.
System.Security.Cryptography.X509Certificates.X509Certificate	GetIssuerName	This method has been deprecated. Use the X509 Certificate.Issuer property instead.
System.Security.Cryptography.X509Certificates.X509Certificate	GetName	This method has been deprecated. Use the X509 Certificate.Subject property instead.
System.Security.Permissions.FileIOPermissionAttribute	All	Use the FileIOPermissionAttribute.ViewAndModify property instead.
System.Security.Permissions.ReflectionPermissionAttribute	ReflectionEmit	This permission is no longer used by the CLR.
System.Security.Permissions.ReflectionPermissionAttribute	TypeInformation	This API has been deprecated.

Type	Member	Message
System.Security.Permissions.ReflectionPermissionFlags	AllFlags	This permission has been deprecated. The AllFlags enumeration member doesn't include RestrictedMemberAccess . Use PermissionState.Unrestricted to get full access.
System.Security.Permissions.ReflectionPermissionFlags	ReflectionEmit	This permission is no longer used by the CLR.
System.Security.Permissions.ReflectionPermissionFlags	TypeInformation	This API has been deprecated.
System.Security.Permissions.RegistryPermissionAttribute	All	Use the RegistryPermissionAttribute.ViewAndModify property instead.
System.Security.Permissions.SecurityAction	Deny	Deny is obsolete and will be removed in a future release of the .NET Framework.
System.Security.Permissions.SecurityAction	RequestMinimum	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.Security.Permissions.SecurityAction	RequestOptional	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.

Type	Member	Message
System.Security.Permissions.SecurityAction	RequestRefuse	Assembly level declarative security is obsolete and is no longer enforced by the CLR by default.
System.Security.PolicyEvidence	AddAssembly	This method is obsolete. Use Evidence.AddAssemblyEvidence instead.
System.Security.PolicyEvidence	AddHost	This method is obsolete. Use Evidence.AddHostEvidence instead.
System.Security.PolicyEvidence	CopyTo	Evidence should not be treated as an System.Collections.ICollection . Use the Evidence.GetHostEnumerator and Evidence.GetAssemblyEnumerator methods rather than using CopyTo .
System.Security.PolicyEvidence	Count	Evidence should not be treated as an System.Collections.ICollection . Use Evidence.GetHostEnumerator and Evidence.GetAssemblyEnumerator to iterate over the evidence to collect a count.
System.Security.PolicyEvidence	Evidence(Object[], Object[])	This constructor is obsolete. Use the Evidence(EvidenceBase[], EvidenceBase[]) constructor instead.
System.Security.PolicyEvidence	GetEnumerator	GetEnumerator is obsolete. Use Evidence.GetHostEnumerator and Evidence.GetAssemblyEnumerator instead.
System.Security.PolicyLevel	AddFullTrustAssembly(StrongName)	Because all GAC assemblies always get full trust, the full trust list is no longer meaningful. You should install any assemblies that are used in security policy in the GAC to ensure they're trusted.
System.Security.PolicyLevel	AddFullTrustAssembly(StrongNameMembershipCondition)	Because all GAC assemblies always get full trust, the full trust list is no longer meaningful. You should install any assemblies that are used in security policy in the GAC to ensure they're trusted.

Type	Member	Message
System.Security.Policy.PolicyLevel	CreateAppDomainLevel	AppDomain policy levels are obsolete and will be removed in a future release of the .NET Framework.
System.Security.Policy.PolicyLevel	FullTrustAssemblies	Because all GAC assemblies always get full trust, the full trust list is no longer meaningful. You should install any assemblies that are used in security policy in the GAC to ensure they're trusted.
System.Security.Policy.PolicyLevel	RemoveFullTrustAssembly(StrongName)	Because all GAC assemblies always get full trust, the full trust list is no longer meaningful. You should install any assemblies that are used in security policy in the GAC to ensure they're trusted.
System.Security.Policy.PolicyLevel	RemoveFullTrustAssembly(StrongNameMembershipCondition)	Because all GAC assemblies always get full trust, the full trust list is no longer meaningful. You should install any assemblies that are used in security policy in the GAC to ensure they're trusted.
System.Threading.Overlapped	EventHandle	This property isn't 64-bit compatible. Use Overlapped.EventHandleIntPtr instead.
System.Threading.Overlapped	Overlapped(Int32, Int32, Int32, IAsyncResult)	This constructor isn't 64-bit compatible. Use the Overlapped(Int32, Int32, IntPtr, IAsyncResult) constructor that takes an System.IntPtr for the event handle.
System.Threading.Overlapped	Pack(ICompletionCallback)	This method isn't safe. Use Overlapped.Pack(ICompletionCallback, Object) instead.
System.Threading.Overlapped	UnsafePack(ICompletionCallback)	This method isn't safe. Use Overlapped.UnsafePack(ICompletionCallback, Object) instead.
System.Threading.Thread	ApartmentState	The ApartmentState property has been deprecated. Use Thread.GetApartmentState , Thread.SetApartmentState or Thread.TrySetApartmentState instead.
System.Threading.Thread	GetCompressedStack	GetCompressedStack is no longer supported. Use the System.Threading.CompressedStack class.

Type	Member	Message
System.Threading.Thread	Resume	Resume has been deprecated. Use other classes in System.Threading , such as Monitor , Mutex , EventWaitHandle , and Semaphore to synchronize threads or protect resources.
System.Threading.Thread	SetCompressedStack	SetCompressedStack is no longer supported. Use the System.Threading.CompressedStack class.
System.Threading.Thread	Suspend	Suspend has been deprecated. Use other classes in System.Threading , such as Monitor , Mutex , EventWaitHandle , and Semaphore , to synchronize threads or protect resources.
System.Threading.ThreadPool	BindHandle(IntPtr)	BindHandle(IntPtr) has been deprecated. Use ThreadPool.BindHandle(SafeHandle) instead.
System.Threading.WaitHandle	Handle	Use the WaitHandle.SafeWaitHandle property instead.

PresentationCore.dll

Type	Member	Message
System.Windows.UIElement	BitmapEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.UIElement	BitmapEffectInput	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.UIElement	PersistId	PersistId is an obsolete property and may be removed in a future release. The value of this property isn't defined.
System.Windows.Media.ContainerVisual	BitmapEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.

Type	Member	Message
System.Windows.Media.ContainerVisual	BitmapEffectInput	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.DrawingContext	PushEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.FormattedText	FormattedText(String, CultureInfo, FlowDirection, Typeface, Double, Brush)	Use the PixelsPerDip override.
System.Windows.Media.FormattedText	FormattedText(String, CultureInfo, FlowDirection, Typeface, Double, Brush, NumberSubstitution)	Use the PixelsPerDip override.
System.Windows.Media.FormattedText	FormattedText(String, CultureInfo, FlowDirection, Typeface, Double, Brush, NumberSubstitution, TextFormattingMode)	Use the PixelsPerDip override.
System.Windows.Media.GlyphRun	GlyphRun()	Use the PixelsPerDip override.
System.Windows.Media.GlyphRun	GlyphRun(GlyphTypeface, Int32, Boolean, Double, IList<UInt16>, Point, IList<Double>, IList<Point>, IList<Char>, String, IList<UInt16>, IList<Boolean>, XmlLanguage)	Use the PixelsPerDip override.
System.Windows.Media.RenderCapability	IsShaderEffectSoftwareRenderingSupported	This property is deprecated. Use the static RenderCapability.IsPixelShaderVersionSupportedInSoftware method instead.
System.Windows.Media.Visual	VisualBitmapEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Visual	VisualBitmapEffectInput	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.

Type	Member	Message
System.Windows.Media.Effects.BevelBitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BevelBitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	CreateBitmapEffectOuter	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	GetOutput	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	InitializeBitmapEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	SetValue	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffectGroup	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BitmapEffectGroup	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.

Type	Member	Message
System.Windows.Media.Effects.BlurBitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.BlurBitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.DropShadowBitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.DropShadowBitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.EmbossBitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.EmbossBitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.OuterGlowBitmapEffect	CreateUnmanagedEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Effects.OuterGlowBitmapEffect	UpdateUnmanagedPropertyState	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.
System.Windows.Media.Media3D.Viewport3DVisual	BitmapEffect	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.

Type	Member	Message
System.Windows.Media.Media3D.Viewport3DVisual	BitmapEffectInput	Bitmap effects are deprecated and no longer function. Consider using System.Windows.Media.Effects.Effect where appropriate instead.

PresentationFramework.dll

Type	Member	Message
System.Windows.Data.BindingListCollectionView	OnBeginChangeLogging	Replaced by CollectionView.OnAllowsCrossThreadChangesChanged .
System.Windows.Data.CollectionView	ClearChangeLog	Replaced by CollectionView.ClearPendingChanges .
System.Windows.Data.CollectionView	OnBeginChangeLogging	Replaced by CollectionView.OnAllowsCrossThreadChangesChanged .
System.Windows.Data.ListCollectionView	OnBeginChangeLogging	Replaced by ListCollectionView.OnAllowsCrossThreadChangesChanged .

System.Activities.dll

Type	Member	Message
System.Activities.Debugger.XamlDebuggerXmlReader	XamlDebuggerXmlReader(XamlReader, IXamlLineInfo, TextReader)	First deprecated in the .NET Framework 4.5. Don't use this constructor. Use XamlDebuggerXmlReader(TextReader) or XamlDebuggerXmlReader(TextReader, XamlSchemaContext) instead.
System.Activities.Debugger.XamlDebuggerXmlReader	XamlDebuggerXmlReader(XamlReader, TextReader)	First deprecated in the .NET Framework 4.5. Don't use this constructor. Use XamlDebuggerXmlReader(TextReader) or XamlDebuggerXmlReader(TextReader, XamlSchemaContext) instead.

System.Activities.Presentation.dll

Type	Member	Message
------	--------	---------

Type	Member	Message
System.Activities.Presentation.DragDropHelper	DoDragMove(WorkflowViewElement, Point)	This method doesn't support dragging multiple items.
System.Activities.Presentation.DragDropHelper	GetCompositeView(DragEventArgs)	First deprecated in the .NET Framework 4.5. This method doesn't support dragging multiple items. Use GetCompositeView(WorkflowViewElement) instead.
System.Activities.Presentation.DragDropHelper	GetDragDropCompletedEffects	This method doesn't support dragging multiple items.
System.Activities.Presentation.DragDropHelper	GetDraggedModelItem	First deprecated in the .NET Framework 4.5. This method doesn't support dragging multiple items. Use GetDraggedModelItems instead.
System.Activities.Presentation.DragDropHelper	GetDroppedObject	First deprecated in the .NET Framework 4.5. This method doesn't support dragging multiple items. Use GetDroppedObjects instead.
System.Activities.Presentation.DragDropHelper	SetDragDropCompletedEffects	This method doesn't support dragging multiple items.
System.Activities.Presentation.Services.ModelChangedEventArgs	ItemsAdded	First deprecated in the .NET Framework 4.5. Don't use this property. Use ModelChangeInfo instead.
System.Activities.Presentation.Services.ModelChangedEventArgs	ItemsRemoved	First deprecated in the .NET Framework 4.5. Don't use this property. Use ModelChangeInfo instead.
System.Activities.Presentation.Services.ModelChangedEventArgs	PropertiesChanged	First deprecated in the .NET Framework 4.5. Don't use this property. Use ModelChangeInfo instead.

System.Core.dll

Type	Member	Message
------	--------	---------

Type	Member	Message
System.Diagnostics.Eventing.Reader.StandardEventKeywords	CorrelationHint	First deprecated in the .NET Framework 4.5.
System.Linq.ParallelEnumerable	Concat<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>)	Incorrect value; use CorrelationHint2 instead.
System.Linq.ParallelEnumerable	Except<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Except<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	GroupJoin<TOuter,TInner,TKey,TResult>(ParallelQuery<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,IEnumerable<TInner>, TResult>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .

Type	Member	Message
System.Linq.ParallelEnumerable	GroupJoin<TOuter,TInner,TKey,TResult>(ParallelQuery<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,IEnumerable<TInner>, TResult>, IEqualityComparer<TKey>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Intersect<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Intersect<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Join<TOuter,TInner,TKey,TResult>(ParallelQuery<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Join<TOuter,TInner,TKey,TResult>(ParallelQuery<TOuter>, IEnumerable<TInner>, Func<TOuter,TKey>, Func<TInner,TKey>, Func<TOuter,TInner,TResult>, IEqualityComparer<TKey>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	SequenceEqual<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .

Type	Member	Message
System.Linq.ParallelEnumerable	SequenceEqual<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Union<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Union<TSource>(ParallelQuery<TSource>, IEnumerable<TSource>, IEqualityComparer<TSource>)	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.ParallelEnumerable	Zip	The second data source of a binary operator must be of type System.Linq.ParallelQuery<TSource> rather than System.Collections.Generic.IEnumerable<T> . To fix this problem, use the AsParallel<TSource>(IEnumerable<TSource>) extension method to convert the right data source to System.Linq.ParallelQuery<TSource> .
System.Linq.Expressions.Expression	Expression(ExpressionType, Type)	Use a different constructor that doesn't take an System.Linq.Expressions.ExpressionType argument. Then override the Expression.NodeType and Expression.Type properties to provide the values that would be specified to this constructor.
System.Linq.Expressions.MemberBinding	MemberBinding	Don't use this constructor. It will be removed in future releases.

Type	Member	Message
System.	AddRule	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	Bind	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	ClearMatch	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	CreateMatchmaker	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	GetCachedRules<T>(RuleCache <T>)	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	GetMatch	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		
System.	GetRuleCache	Use of this member generates a compiler error.
Runtim		
e.Comp		Don't use this method.
ilerServi		
ces.Call		
SiteOps		

Type	Member	Message
System.Runtime.CompilerServices.CallSiteOps	GetRules	Use of this member generates a compiler error.
System.Runtime.CompilerServices.CallSiteOps	MoveRule<T>(RuleCache<T>, T, Int32)	Don't use this method.
System.Runtime.CompilerServices.CallSiteOps	SetNotMatched	Use of this member generates a compiler error.
System.Runtime.CompilerServices.CallSiteOps	UpdateRules	Don't use this method.
System.Runtime.CompilerServices.RunTimeOpS	CreateRuntimeVariables()	Use of this member generates a compiler error.
System.Runtime.CompilerServices.RunTimeOpS	CreateRuntimeVariables(Object[], Int64[])	Don't use this method.

Type	Member	Message
System. Runtim e.Comp ilerServi ces.Run timeOps	ExpandoCheckVersion	Use of this member generates a compiler error. Don't use this method.
System. Runtim e.Comp ilerServi ces.Run timeOps	ExpandoPromoteClass	Use of this member generates a compiler error. Don't use this method.
System. Runtim e.Comp ilerServi ces.Run timeOps	ExpandoTryDeleteValue	Use of this member generates a compiler error. Don't use this method.
System. Runtim e.Comp ilerServi ces.Run timeOps	ExpandoTryGetValue	Use of this member generates a compiler error. Don't use this method.
System. Runtim e.Comp ilerServi ces.Run timeOps	ExpandoTrySetValue	Use of this member generates a compiler error. Don't use this method.
System. Runtim e.Comp ilerServi ces.Run timeOps	MergeRuntimeVariables	Use of this member generates a compiler error. Don't use this method.

Type	Member	Message
System.Runtime.CompilerServices.RunTimeOptions	Quote	Use of this member generates a compiler error.

System.Data.dll

Type	Member	Message
System.Data.DataSysDescriptionAttribute	DataSysDescriptionAttribute	DataSysDescriptionAttribute has been deprecated.
System.Data.Common.DataAdapter	CloneInternals	CloneInternals has been deprecated. Use the DataAdapter (DataAdapter) constructor.
System.Data.Common.DBDataPermission	DBDataPermission()	Use of this member generates a compiler error. This constructor has been deprecated. Pass the DBDataPermission(PermissionState) constructor a value of PermissionState.None .
System.Data.Common.DBDataPermission	DBDataPermission(PermissionState, Boolean)	Use of this member generates a compiler error. This constructor has been deprecated. Pass the DBDataPermission(PermissionState) constructor a value of PermissionState.None .
System.Data.OdbcParameterCollection	Add(String, Object)	Add(String, Object) has been deprecated. Use OdbcParameterCollection.AddWithValue(String, Object) .
System.Data.OdbcPermission	OdbcPermission()	Use of this member generates a compiler error. OdbcPermission() has been deprecated. Pass the OdbcPermission(PermissionState) constructor a value of PermissionState.None .

Type	Member	Message
System.Data.Odbc.OdbcPermission	OdbcPermission(PermissionState, Boolean)	Use of this member generates a compiler error. OdbcPermission(PermissionState, Boolean) has been deprecated. Pass the OdbcPermission(PermissionState) constructor a value of PermissionState.None.
System.Data.OleDb.OleDbParameterCollection	Add(String, Object)	Add(String, Object) has been deprecated. Use the OleDbParameterCollection.AddWithValue method.
System.Data.OleDb.OleDbPermission	OleDbPermission()	Use of this member generates a compiler error. OleDbPermission() has been deprecated. Pass the OleDbPermission(PermissionState) a value of PermissionState.None.
System.Data.OleDb.OleDbPermission	OleDbPermission(PermissionState, Boolean)	Use of this member generates a compiler error. OleDbPermission(PermissionState, Boolean) has been deprecated. Pass the OleDbPermission(PermissionState) a value of PermissionState.None.
System.Data.OleDb.OleDbPermission	Provider	The OleDbPermission.Provider property has been deprecated. Use the DBDataPermission.Add(String, String, KeyRestrictionBehavior) method.
System.Data.OleDb.OleDbPermissionAttribute	Provider	The OleDbPermissionAttribute.Provider property has been deprecated. Use the DBDataPermission.Add(String, String, KeyRestrictionBehavior) method.
System.Data.SqlClient.SqlClientPermission	SqlClientPermission()	Use of this constructor generates a compiler error. SqlClientPermission() has been deprecated. Pass the SqlClientPermission(PermissionState) constructor a value of PermissionState.None.
System.Data.SqlClient.SqlClientPermission	SqlClientPermission(PermissionState, Boolean)	Use of this constructor generates a compiler error. SqlClientPermission(PermissionState, Boolean) has been deprecated. Pass the SqlClientPermission(PermissionState) constructor a value of PermissionState.None.
System.Data.SqlClient.SqlClientConnectionStringBuilder	ConnectionReset	ConnectionReset has been deprecated. System.Data.SqlClient.SqlConnection will ignore the 'connection reset' keyword and always reset the connection.
System.Data.SqlClient.SqlClientParameterCollection	Add(String, Object)	Add(String, Object) has been deprecated. Use SqlParameterCollection.AddWithValue.

System.Data.Entity.dll

Type	Member	Message
System.Data.Metadata.Edm.AssociationSetEnd	Role	This property is going away, Use the AssociationSetName property instead.
System.Data.Metadata.Edm.MetadataWorkspace	GetRequiredOriginalValueMembers	First deprecated in the .NET Framework 4.5. Use MetadataWorkspace.GetRelevantMembersForUpdate instead.
System.Data.Objects.ObjectContext	ApplyPropertyChanges	Use ObjectContext.ApplyCurrentValues instead.
System.Data.Objects.ObjectContext	SaveChanges(Boolean)	Use SaveChanges(SaveOptions) instead.

System.Data.OracleClient.dll

Type	Member	Message
System.Data.OracleClient.OracleParameter	Precision	Precision has been deprecated. Use the System.Math classes to explicitly set the precision of a decimal.
System.Data.OracleClient.OracleParameter	Scale	Scale has been deprecated. Use the System.Math classes to explicitly set the scale of a decimal.
System.Data.OracleClient.OracleParameterCollection	Add(String, Object)	Add(String, Object) has been deprecated. Use OracleParameterCollection.AddWithValue .

System.Design.dll

Type	Member	Message
System.ComponentModel.Design.ComponentDesigner	InitializeNonDefault	This method has been deprecated. Use ComponentDesigner.InitializeExistingComponent instead.

Type	Member	Message
System.ComponentMode.Design.ComponentDesigner	OnSetComponentDefaults	This method has been deprecated. Use ComponentDesigner.InitializeNewComponent instead.
System.ComponentMode.Design.DesignSurface	CreateComponent	The CreateComponent method has been replaced by CreateInstance(T) .
System.ComponentMode.Design.Serialization.CodeDomSerializer	SerializeToReferenceExpression	This method has been deprecated. Use SerializeToExpression or GetExpression instead.
System.Web.UI.Design.ControlDesigner	DesignTimeElementView	<p>Use of this property generates a compiler error.</p> <p>Error: This property can no longer be referenced, and is included to support existing compiled applications. The design-time element view architecture is no longer used.</p>
System.Web.UI.Design.ControlDesigner	DesignTimeHtmlRequiresLoadComplete	The recommended alternative is to use ControlDesigner.SetViewFlags(ViewFlags.DesignTimeHtmlRequiresLoadComplete, true) .
System.Web.UI.Design.ControlDesigner	GetPersistenceInnerHtml	The recommended alternative is ControlDesigner.GetPersistenceContent .
System.Web.UI.Design.ControlDesigner	IsDirty	The recommended alternative is to use ControlDesigner.Tag.SetDirty and ControlDesigner.Tag.IsDirty .
System.Web.UI.Design.ControlDesigner	IsPropertyBound	The recommended alternative is ControlDesigner.DataBindings.Containers . The System.Web.UI.DataBindingCollection class allows more control of the data bindings associated with the control.

Type	Member	Message
System.Web.UI.Design.ContainerDesigner	OnBindingsCollectionChanged	The recommended alternative is to handle the <code>ControlDesigner.DataBindings.Changed</code> event. The <code>DataBindingCollection</code> collection returned by the <code>ControlDesigner.DataBindings</code> property allows more control of the data bindings associated with the control.
System.Web.UI.Design.ContainerDesigner	OnControlResize	The recommended alternative is <code>OnComponentChanged</code> , which is called when any property of the control is changed.
System.Web.UI.Design.ContainerDesigner	RaiseResizeEvent	Use of this method isn't recommended because resizing is handled by the <code>ControlDesigner.OnComponentChanged</code> method.
System.Web.UI.Design.ContainerDesigner	ReadOnly	The recommended alternative is to inherit from <code>System.Web.UI.Design.ContainerControlDesigner</code> instead and to use an <code>System.Web.UI.Design.EditableDesignerRegion</code> . Regions allow for better control of the content in the designer.
System.Web.UI.Design.HtmlControlDesigner	Behavior	The recommended alternative is <code>ControlDesigner.Tag</code> .
System.Web.UI.Design.HtmlControlDesigner	DesignTimeElement	<p>Use of this property generates a compiler error.</p> <p>Error: This property can no longer be referenced, and is included to support existing compiled applications. The design-time element may not always provide access to the element in the markup. There are alternate methods on <code>System.Web.UI.Design.WebFormsRootDesigner</code> for handling client script and controls.</p>
System.Web.UI.Design.HtmlControlDesigner	OnBehaviorAttached	The recommended alternative is <code>ControlDesigner.Tag</code> .
System.Web.UI.Design.HtmlControlDesigner	OnBehaviorDetaching	The recommended alternative is <code>ControlDesigner.Tag</code> .
System.Web.UI.Design.HtmlControlDesigner	OnBindingsCollectionChanged	The recommended alternative is to handle the <code>HtmlControlDesigner.DataBindings.Changed</code> event. The <code>DataBindingCollection</code> collection returned by the <code>HtmlControlDesigner.DataBindings</code> property allows more control of the data bindings associated with the control.

Type	Member	Message
System.Web.UI.Design.HtmlControlDesigner	ShouldCodeSerialize	Use of this property isn't recommended because code serialization isn't supported.
System.Web.UI.Design.TemplatePlatedControlDesigner	ActiveTemplateEditingFrame	Use of this property isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	CreateTemplateEditingFrame	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	EnterTemplateMode	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	ExitTemplateMode	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	GetCachedTemplateEditingVerbs	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	GetTemplateContainerDataItemProperty	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplatePlatedControlDesigner	GetTemplateContainerDataSource	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .

Type	Member	Message
System.Web.UI.Design.TemplateControlDesigner	GetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplateControlDesigner	GetTemplateEditingVerbs	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplateControlDesigner	GetPropertyOfType	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.TemplateControlDesigner	InTemplateMode	The recommended alternative is ControlDesigner.InTemplateMode .
System.Web.UI.Design.TemplateControlDesigner	OnBehaviorAttached	The recommended alternative is ControlDesigner.Tag .
System.Web.UI.Design.TemplateControlDesigner	SetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.BaseDataListDesigner	GetTemplateContainerDataSource	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.BaseDataListDesigner	OnAutoFormat	Use of this method isn't recommended because the AutoFormat dialog is launched by the designer host. The list of available AutoFormats is exposed by the ControlDesigner.AutoFormats property.

Type	Member	Message
System.Web.UI.Design.WebControls.DataGridDesigner	CreateTemplateEditingFrame	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataGridDesigner	GetCachedTemplateEditingVerbs	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataGridDesigner	GetTemplateContainerDataItemProperty	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataGridDesigner	GetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataGridDesigner	GetTemplatePropertyParentType	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataGridDesigner	SetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataListDesigner	CreateTemplateEditingFrame	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataListDesigner	GetCachedTemplateEditingVerbs	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .

Type	Member	Message
System.Web.UI.Design.WebControls.DataListDesigner	GetTemplateContentDataItemProperty	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataListDesigner	GetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.DataListDesigner	SetTemplateContent	Use of this method isn't recommended because template editing is handled in System.Web.UI.Design.ControlDesigner . To support template editing, expose template data in the ControlDesigner.TemplateGroups property and call ControlDesigner.SetViewFlags(ViewFlags.TemplateEditing, true) .
System.Web.UI.Design.WebControls.PanelDesigner	OnBehaviorAttached	The recommended alternative is ControlDesigner.Tag .
System.Windows.Forms.Design.ControlDesigner	OnSetComponentDefaults	This method has been deprecated. Use ControlDesigner.InitializeNewComponent instead.

System.dll

Type	Member	Message
Microsoft.CSharp.CodeProvider	CreateCompiler	Callers should not use the System.CodeDom.Compiler.ICodeCompiler interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class.
Microsoft.CSharp.CodeProvider	CreateGenerator	Callers should not use the System.CodeDom.Compiler.ICodeGenerator interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class.
Microsoft.VisualBasic.CodeProvider	CreateCompiler	Callers should not use the System.CodeDom.Compiler.ICodeCompiler interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class.

Type	Member	Message
Microsoft.VisualBasic.CodeProvider	CreateGenerator	Callers should not use the System.CodeDom.Compiler.ICodeGenerator interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class.
Microsoft.Win32.SystemEvents	LowMemory	This event has been deprecated.
System.Uri	Canonicalize	The method has been deprecated. It isn't used by the system.
System.Uri	CheckSecurity	The method has been deprecated. It isn't used by the system.
System.Uri	Escape	The method has been deprecated. It isn't used by the system.
System.Uri	EscapeString	The method has been deprecated. Use the GetComponents method or the static EscapeUriString method to escape a Uri component or a string.
System.Uri	IsBadFilesystemCharacter	The method has been deprecated. It isn't used by the system.
System.Uri	IsExcludedCharacter	The method has been deprecated. It isn't used by the system.
System.Uri	IsReservedCharacter	The method has been deprecated. It isn't used by the system.
System.Uri	MakeRelative	The method has been deprecated. Use Uri.MakeRelativeUri .
System.Uri	Parse	The method has been deprecated. It isn't used by the system.
System.Uri	Unescape	The method has been deprecated. Use the Uri.GetComponents method or the static Uri.EscapeUriString method to escape a Uri component or a string.
System.Uri	Uri(String, Boolean)	The constructor has been deprecated. Use Uri(String) . The <code>dontEscape</code> parameter is deprecated and is always <code>false</code> .
System.Uri	Uri(Uri, String, Boolean)	The constructor has been deprecated. Use Uri(Uri, String) . The <code>dontEscape</code> parameter is deprecated and is always <code>false</code> .

Type	Member	Message
System.CodeDom.Compiler.CodeDomProvider	CreateCompiler	Callers should not use the System.CodeDom.Compiler.ICodeCompiler interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class. Those inheriting from CodeDomProvider must still implement this interface, and should exclude this warning or also obsolete this method.
System.CodeDom.Compiler.CodeDomProvider	CreateGenerator	Callers should not use the System.CodeDom.Compiler.ICodeGenerator interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class. Those inheriting from CodeDomProvider must still implement this interface, and should exclude this warning or also obsolete this method.
System.CodeDom.Compiler.CodeDomProvider	CreateParser	Callers should not use the System.CodeDom.Compiler.ICodeParser interface and should instead use the methods directly on the System.CodeDom.Compiler.CodeDomProvider class. Those inheriting from CodeDomProvider must still implement this interface, and should exclude this warning or also obsolete this method.
System.CodeDom.Compiler.CompilerParameters	Evidence	CAS policy is obsolete and will be removed in a future release of the .NET Framework. For more information, see Security Changes in the .NET Framework 4 .
System.CodeDom.Compiler.CompilerResults	Evidence	CAS policy is obsolete and will be removed in a future release of the .NET Framework. For more information, see Security Changes in the .NET Framework 4 .
System.Collections.Specialized.NameObjectCollectionBase	NameObjectCollectionBase(IEqualityComparer)	Use NameObjectCollectionBase(IEqualityComparer) instead.
System.Collections.Specialized.NameObjectCollectionBase	NameObjectCollectionBase(Int32, IEqualityComparer)	Use NameObjectCollectionBase(Int32, IEqualityComparer) instead.
System.Collections.Specialized.NameValueCollection	NameValueCollection(IEqualityComparer)	Use NameValuePairCollection(IEqualityComparer) instead.

Type	Member	Message
System.Collections.Specialized.NameValueCollection	NameValueCollection(Int32, IEqualityComparer)	Use NameValuePairCollection(Int32, IEqualityComparer) instead.
System.ComponentModel.AsyncCompletedEventArgs	AsyncCompletedEventArgs()	First deprecated in the .NET Framework 4.5. Use of this member generates a compiler error. This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.
System.ComponentModel.MemberDescriptor	GetInvokee	This method has been deprecated. Use MemberDescriptor.GetInvocationTarget instead.
System.ComponentModel.TypeDescriptor	ComNativeDescriptorHandler	This property has been deprecated. Use a type description provider to supply type information for COM types instead.
System.ComponentModel.Design.DesignerTransactionCloseEventArgs	DesignerTransactionCloseEventArgs(Boolean, Boolean)	This constructor is obsolete. Use DesignerTransactionCloseEventArgs(Boolean, Boolean) instead.
System.ComponentModel.Design.SelectionTypes	Click	This value has been deprecated. Use SelectionTypes.Primary instead.
System.ComponentModel.Design.SelectionTypes	MouseDown	This value has been deprecated. It is no longer supported.
System.ComponentModel.Design.SelectionTypes	MouseUp	This value has been deprecated. It is no longer supported.

Type	Member	Message
System.ComponentModel.Design.SelectionTypes	Normal	This value has been deprecated. Use SelectionTypes.Auto instead.
System.ComponentModel.Design.SelectionTypes	Valid	This value has been deprecated. Use System.Enum class methods to determine valid values, or use a type converter.
System.ComponentModel.Design.ViewTechnology	Passthrough	This value has been deprecated. Use ViewTechnology.Default instead.
System.ComponentModel.Design.ViewTechnology	WindowsForms	This value has been deprecated. Use ViewTechnology.Default instead.
System.Configuration.ConfigurationException	ConfigurationException()	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	ConfigurationException(String)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	ConfigurationException(String, Exception)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	ConfigurationException(String, Exception, String, Int32)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	ConfigurationException(String, Exception, XmlNode)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .

Type	Member	Message
System.Configuration.ConfigurationException	ConfigurationException(String, String, Int32)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	ConfigurationException(String, XmlNode)	This class is obsolete. To create a new exception, create a System.Configuration.ConfigurationErrorsException .
System.Configuration.ConfigurationException	GetXmlNodeName	This class is obsolete. Use ConfigurationErrorsException.GetFilename instead.
System.Configuration.ConfigurationException	GetXmlNodeLineNumber	This class is obsolete. Use ConfigurationErrorsException.GetLineNumber(XmlNode) instead
System.Configuration.ConfigurationSettings	AppSettings	This method is obsolete. It has been replaced by ConfigurationManager.AppSettings .
System.Configuration.ConfigurationSettings	GetConfig	This method is obsolete. It has been replaced by ConfigurationManager.GetSection .
System.Diagnostics.EventLog	CreateEventSource(String, String, String)	This method has been deprecated. Use CreateEventSource(EventSourceCreationData) instead.
System.Diagnostics.EventLogEntry	EventID	This property has been deprecated. Use EventLogEntry.InstanceId instead.
System.Diagnostics.EventLogPermissionAccess	Audit	This member has been deprecated. Use EventLogPermissionAccess.Administer instead.
System.Diagnostics.EventLogPermissionAccess	Browse	This member has been deprecated. Use EventLogPermissionAccess.Administer instead.

Type	Member	Message
System.Diagnostics.EventLogPermissionAccess	Instrument	This member has been deprecated. Use EventLogPermissionAccess.Write instead.
System.Diagnostics.InstanceDataCollection	InstanceDataCollection	This constructor has been deprecated. Use InstanceDataCollectionCollection.Item[] to get an instance of this collection instead.
System.Diagnostics.InstanceDataCollectionCollection	InstanceDataCollection	This constructor has been deprecated. Use PerformanceCounterCategory.ReadCategory to get an instance of this collection instead.
System.Diagnostics.PerformanceCounter	DefaultFileMappingSize	This field has been deprecated and isn't used. Use machine.config or an application configuration file to set the size of the System.Diagnostics.PerformanceCounter file mapping.
System.Diagnostics.PerformanceCounterCategory	Create(String, String, CounterCreationDataCollection)	This method has been deprecated. Use Create(String, String, PerformanceCounterCategoryType, CounterCreationDataCollection) instead.
System.Diagnostics.PerformanceCounterCategory	Create(String, String, String, String)	This method has been deprecated. Use Create(String, String, PerformanceCounterCategoryType, String, String) instead.
System.Diagnostics.PerformanceCounterManager	ICollectData.CloseData	This class has been deprecated. Use the performance counters through the PerformanceCounter class instead.
System.Diagnostics.PerformanceCounterManager	ICollectData.CollectData	This class has been deprecated. Use the performance counters through the PerformanceCounter class instead.
System.Diagnostics.PerformanceCounterManager	PerformanceCounterManager	This class has been deprecated. Use the performance counters through the System.Diagnostics.PerformanceCounter class instead.

Type	Member	Message
System.Diagnostics.PerformanceCounterPermissionAccess	Browse	This member has been deprecated. Use PerformanceCounterPermissionAccess.Read instead.
System.Diagnostics.PerformanceCounterPermissionAccess	Instrument	This member has been deprecated. Use PerformanceCounterPermissionAccess.Write instead.
System.Diagnostics.Process	NonpagedSystemMemorySize	This property has been deprecated. Use Process.NonpagedSystemMemorySize64 instead.
System.Diagnostics.Process	PagedMemorySize	This property has been deprecated. Use PagedMemorySize64 instead.
System.Diagnostics.Process	PagedSystemMemorySize	This property has been deprecated. Use Process.PagedSystemMemorySize64 instead.
System.Diagnostics.Process	PeakPagedMemorySize	This property has been deprecated. Use Process.PeakPagedMemorySize64 instead.
System.Diagnostics.Process	PeakVirtualMemorySize	This property has been deprecated. Use Process.PeakVirtualMemorySize64 instead.
System.Diagnostics.Process	PeakWorkingSet	This property has been deprecated. Use Process.PeakWorkingSet64 instead.
System.Diagnostics.Process	PrivateMemorySize	This property has been deprecated. Use Process.PrivateMemorySize64 instead.
System.Diagnostics.Process	VirtualMemorySize	This property has been deprecated. Use Process.VirtualMemorySize64 instead.
System.Diagnostics.Process	WorkingSet	This property has been deprecated. Use Process.WorkingSet64 instead.

Type	Member	Message
System.Net.Dns	BeginGetHostByName	BeginGetHostName is obsolete for this type, Use Dns.BeginGetHostEntry(String, AsyncCallback, Object) instead.
System.Net.Dns	BeginResolve(String, AsyncCallback, Object)	BeginResolve(String, AsyncCallback, Object) is obsolete for this type, Use Dns.BeginGetHostEntry(String, AsyncCallback, Object) instead.
System.Net.Dns	EndGetHostByName	EndGetHostName is obsolete for this type, Use Dns.EndGetHostEntry instead.
System.Net.Dns	EndResolve	EndResolve is obsolete for this type, Use Dns.EndGetHostEntry instead.
System.Net.Dns	GetHostByAddress(IPAddress)	GetHostByAddress(IPAddress) is obsolete for this type, Use Dns.GetHostEntry(IPAddress) instead.
System.Net.Dns	GetHostByAddress(String)	GetHostByAddress(String) is obsolete for this type, Use Dns.GetHostEntry(String) instead.
System.Net.Dns	GetHostByName	GetHostName is obsolete for this type, Use Dns.GetHostEntry(String) instead.
System.Net.Dns	Resolve	Resolve is obsolete for this type, Use Dns.GetHostEntry(String) instead.
System.Net.FileWebRequest	FileWebRequest	Serialization is obsolete for this type.
System.Net.FileWebResponse	FileWebResponse	Serialization is obsolete for this type.
System.Net.HttpWebRequest	HttpWebRequest()	First deprecated in the .NET Framework 4.5. Use of this member generates a compiler error. This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.
System.Net.HttpWebRequest	HttpWebRequest(SerializationInfo, StreamingContext)	Serialization is obsolete for this type.

Type	Member	Message
System.Net.HttpWebResponse	HttpWebResponse()	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net.HttpWebResponse	HttpWebResponse(SerializationInfo, StreamingContext)	Serialization is obsolete for this type.
System.Net.IPEndPoint	Address	<p>This property has been deprecated. It is address family dependent.</p> <p>Use the IPAddress.Equals method to perform comparisons instead.</p>
System.Net.ServicePointManager	CertificatePolicy	CertificatePolicy is obsolete for this type. Use ServicePointManager.ServerCertificateValidationCallback instead.
System.Net.WebClient	AllowReadStreamBuffering	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net.WebClient	AllowWriteStreamBuffering	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net.WebClient	System.Net.WebClient.OnWriteStreamClosed	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net.WebClient	WriteStreamClosed	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>

Type	Member	Message
System.Net.WebProxy	GetDefaultProxy	This method has been deprecated. Use the proxy selected for you by default.
System.Net.WebRequest	CreatorInstance	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net.WebRequest	RegisterPort ableWebRequestCreate or	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net. WriteStream ClosedEvent Args	Error	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net. WriteStream ClosedEvent Args	WriteStrea mClosedEve ntArgs	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net. Mail.MailMe ssage	ReplyTo	<p>ReplyTo is obsolete for this type. Use MailMessage.ReplyToList instead, which can accept multiple addresses.</p>
System.Net. NetworkInfo rmation.Net workChange	NetworkCh ange()	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>
System.Net. NetworkInfo rmation.Net workChange	RegisterNet workChang e(NetworkC hange)	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.</p>

Type	Member	Message
System.Net. Sockets.Soc ket	SupportsIPv 4	SupportsIPv4 is obsolete for this type. Use Socket.OSSupportsIPv4 instead.
System.Net. Sockets.Soc ket	SupportsIPv 6	SupportsIPv6 is obsolete for this type. Use Socket.OSSupportsIPv6 instead.
System.Net. Sockets.Soc ketAsyncEve ntArgs	SocketClien tAccessPolic yProtocol	First deprecated in the .NET Framework 4.5. Use of this member generates a compiler error. This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.
System.Net. Sockets.TcpL istener	TcpListener (Int32)	This method has been deprecated. Use TcpListener(IPAddress, Int32) instead.
System.Net. WebSockets. WebSocket	IsApplicatio nTargeting4 5	First deprecated in the .NET Framework 4.5. This member is for internal use only and will be removed in a future version of the .NET Framework. Don't call it.
System.Secu rity.Claims.D ynamicRole ClaimProvid er	AddDynam icRoleClaims	First deprecated in the .NET Framework 4.5. Use of this member generates a compiler error. Use System.Security.Claims.AuthenticationManager to add claims to a ClaimsIdentity .

System.Drawing.dll

Type	Member	Message
System.Drawing.Fo ntFamily	GetFamilies	Don't use the GetFamilies method; use the FontFamil y.Families property instead.
System.Drawing.Im aging.EncoderPara meter	EncoderParameter(E ncoder, Int32, Int32, Int32)	First deprecated in the .NET Framework 4.5. This constructor has been deprecated. Use EncoderP arameter(Encoder, Int32, EncoderParameterValueTyp e, IntPtr) .

System.Messaging.dll

Type	Member	Message
System.Messaging.MessageQueue	GetEnumerator	This method returns a System.Messaging.MessageEnumerator that implements the MessageEnumerator.RemoveCurrent family of methods incorrectly. Use MessageQueue.GetMessageEnumerator2 instead.
System.Messaging.MessageQueue	GetMessageEnumerator	This method returns a System.Messaging.MessageEnumerator that implements the MessageEnumerator.RemoveCurrent family of methods incorrectly. Use MessageQueue.GetMessageEnumerator2 instead.

System.ServiceModel.dll

Type	Member	Message
System.ServiceModel.BasicHttpBinding	EnableHttpGetCookieContainer	First deprecated in the .NET Framework 4.5.
	Containerr	This property is obsolete. To enable Http CookieContainer , use the HttpBindingBase.AllowCookies property instead.
System.ServiceModel.Configuration.BindingSection	NetPeerTcpBinding	First deprecated in the .NET Framework 4.5. The peer channel feature is obsolete and will be removed in the future.
System.ServiceModel.Dispatcher.ClientOperationCompatBase	ParameterInspector	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is n't intended to be used directly from your code.
System.ServiceModel.Dispatcher.ClientRuntimeCompatBase	MessageInspector	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is n't intended to be used directly from your code.
System.ServiceModel.Dispatcher.ClientRuntimeCompatBase	Operations	First deprecated in the .NET Framework 4.5. Use of this type generates a compiler error. This API supports the .NET Framework infrastructure and is n't intended to be used directly from your code.

Type	Member	Message
System.ServiceModel.Securit y.WindowsClientCredential	AllowNtLM	This property is deprecated and is maintained for backward compatibility only. The local machine policy will be used to determine if NTLM should be used.

System.ServiceModel.Discovery.dll

Type	Member	Message
System.ServiceModel.Disco very.UdpAnnounce mentEndpoint	Transpor tSettings	First deprecated in the .NET Framework 4.5. The TransportSettings property is obsolete. Consider using System.ServiceModel.Channels.UdpTransportBindingElement for setting the transport properties.
System.ServiceModel.Disco very.UdpDiscoveryE ndpoint	Transpor tSettings	First deprecated in the .NET Framework 4.5. The TransportSettings property is obsolete. Consider using System.ServiceModel.Channels.UdpTransportBindingElement for setting the transport properties.

System.Web.DataVisualization.dll

Type	Member	Message
System.Web.UI.DataVisualization.Charting.Chart	ViewStateData	ViewStateData has been deprecated. Investigate Control.ViewState instead.

System.Web.dll

Type	Member	Message
System.Web.HttpContext	GetAppConfig	The recommended alternative is WebConfigurationManager.Get WebApplicationSection in System.Web.dll.
System.Web.HttpContext	GetConfig	The recommended alternative is HttpContext.GetSection in System.Web.dll.
System.Web.HttpUtility	UrlEncodeU nicode	First deprecated in the .NET Framework 4.5. This method produces non-standards-compliant output and has interoperability issues. The preferred alternative is UrlEncode(String) .

Type	Member	Message
System.Web.HttpUtility	UrlEncodeUnicodeToBytes	<p>First deprecated in the .NET Framework 4.5.</p> <p>This method produces non-standards-compliant output and has interoperability issues. The preferred alternative is UrlEncodeToBytes(String).</p>
System.Web.Configuration.AuthenticationModule	Passport	This field is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.Configuration.AuthenticationSection	Passport	This property is obsolete. The Passport authentication product is no longer supported and has been superseded by Microsoft Account .
System.Web.Configuration.HttpCapabilitiesBase	JavaScript	The recommended alternative is the HttpCapabilitiesBase.EcmaScriptVersion property. A Version.Major version value greater than or equal to 1 implies JavaScript support.
System.Web.Configuration.SystemWebSectionGroup	MobileControls	System.Web.Mobile.dll is obsolete.
System.Web.Routing.UrlRoutingModule	PostMapRequestHandler	This method is obsolete. Override the Init method to use the PostMapRequestHandler event.
System.Web.Security.FormsAuthentication	Authenticate	<p>First deprecated in the .NET Framework 4.5.</p> <p>The recommended alternative is to use the System.Web.Security.Membership APIs, such as Membership.ValidateUser.</p>
System.Web.Security.FormsAuthentication	HashPasswordForStoringInConfigFile	<p>First deprecated in the .NET Framework 4.5.</p> <p>The recommended alternative is to use the System.Web.Security.Membership APIs, such as Membership.CreateUser.</p>
System.Web.Security.MachineKey	Decode	<p>First deprecated in the .NET Framework 4.5.</p> <p>This method is obsolete and is only provided for compatibility with existing code. It is recommended that new code use the Protect and Unprotect methods instead.</p>

Type	Member	Message
System.Web.Security.MachineKey	Encode	First deprecated in the .NET Framework 4.5.
		This method is obsolete and is only provided for compatibility with existing code. It is recommended that new code use the Protect and Unprotect methods instead.
System.Web.UI.Page	FileDependencies	The recommended alternative is HttpResponse.AddFileDependencies .
System.Web.UI.Page	GetPostBackClientEvent	The recommended alternative is ClientScriptManager.GetPostBackEventReference .
System.Web.UI.Page	GetPostBackClientHyperlink	The recommended alternative is ClientScriptManager.GetPostBackClientHyperlink .
System.Web.UI.Page	GetPostBackEventReference(Control)	The recommended alternative is ClientScriptManager.GetPostBackEventReference(Control, String) .
System.Web.UI.Page	GetPostBackEventReference(Control, String)	The recommended alternative is ClientScriptManager.GetPostBackEventReference .
System.Web.UI.Page	IsClientScriptBlockRegistered	The recommended alternative is ClientScriptManager.IsClientScriptBlockRegistered .
System.Web.UI.Page	IsStartupScriptRegistered	The recommended alternative is ClientScriptManager.IsStartupScriptRegistered .
System.Web.UI.Page	RegisterArrayDeclaration	The recommended alternative is ClientScriptManager.RegisterArrayDeclaration .
System.Web.UI.Page	RegisterClientScriptBlock	The recommended alternative is ClientScriptManager.RegisterClientScriptBlock .
System.Web.UI.Page	RegisterHiddenField	The recommended alternative is ClientScriptManager.RegisterHiddenField .
System.Web.UI.Page	RegisterOnSubmitStatement	The recommended alternative is ClientScriptManager.RegisterOnSubmitStatement .
System.Web.UI.Page	RegisterStartupScript	The recommended alternative is ClientScriptManager.RegisterStartupScript .

Type	Member	Message
System.Web.UI.Page	SmartNavigation	The recommended alternative is Page.SetFocus and Page.MaintainScrollPositionOnPostBack .
System.Web.UI.TemplateControl	AutoHandlers	Use of this property isn't recommended because it is no longer useful.
System.Web.UI.WebControls.GridView	CreateAutoGeneratedColumn	<p>First deprecated in the .NET Framework 4.5.</p> <p>This method is kept for backward compatibility. This API is no longer used.</p>
System.Web.UI.WebControls.Xml	Document	The recommended alternative is the Xml.XPathNavigator property. Create a System.Xml.XPath.XPathDocument and call XPathDocument.CreateNavigator to create an System.Xml.XPath.XPathNavigator .

System.Web.DynamicData.dll

Type	Member	Message
System.Web.DynamicData.DynamicDataExtensions	EnablePersistedSelection	Use the <code>EnablePersistedSelection</code> property on a databound control such as System.Web.UI.WebControls.GridView or System.Web.UI.WebControls.ListView .

System.Web.Extensions.dll

Type	Member	Message
System.Web.UI.CompositeScriptReference	IsFromSystemWebExtensions	Use CompositeScriptReference.IsAjaxFrameworkScript .
System.Web.UI.ScriptManager	ScriptPath	This property is obsolete. Set the System.Web.UI.ScriptReference.Path property on each individual System.Web.UI.ScriptReference instead.
System.Web.UI.ScriptReference	IgnoreScriptPath	This property is obsolete. Instead of using ScriptManager.ScriptPath , set the System.Web.UI.ScriptReference.Path property on each individual System.Web.UI.ScriptReference .
System.Web.UI.ScriptReference	IsFromSystemWebExtensions	Use ScriptReference.IsAjaxFrameworkScript .

Type	Member	Message
System.Web.U I.ScriptReferen ceBase	IsFromSys temWebE xtensions	Use ScriptReferenceBase.IsAjaxFrameworkScript .
System.Web.U I.ScriptReferen ceBase	NotifyScri ptLoaded	NotifyScriptLoaded is no longer required in script references.
System.Web.U I.ScriptResourc eAttribute	ScriptRes ourceNa me	This property is obsolete. Use ScriptResourceAttribute.StringResourc eName instead.
System.Web.U I.ScriptResourc eAttribute	TypeNam e	This property is obsolete. Use ScriptResourceAttribute.StringResourc eClientTypeName instead.

System.Web.Services.dll

Type	Member	Message
System.Web.Services.Disc overy.DiscoveryClientPro tocol	LoadExte rnals	This method will be removed from a future version. The m ethod call is no longer required for resource discovery.
System.Web.Services.Prot ocols.SoapHeaderAttribu te	Required	This property will be removed from a future version. The p resence of a particular header in a SOAP message is no lo nger enforced.

System.Windows.Forms.dll

Type	Member	Message
System.Windows.Forms. AccessibleStates	Valid	This enumeration value has been deprecated. There is no r eplacement.
System.Windows.Forms. ComboBox	AddItems Core	This method has been deprecated. There is no replacemen t.
System.Windows.Forms. Control	RenderRig htToLeft	This property has been deprecated. Use Control.RightToLe ft instead.
System.Windows.Forms. Control	Scale(Sing le)	This method has been deprecated. Use the Control.Scale(S izeF) method instead.

Type	Member	Message
System.Windows.Forms. Control	Scale(Sing le, Single)	This method has been deprecated. Use the Control.Scale(S izeF) method instead.
System.Windows.Forms. Form	ApplyAut oScaling	This method has been deprecated. Use the PerformAutoSc ale method instead.
System.Windows.Forms. Form	AutoSize	This property has been deprecated. Use the ContainerCon trol.AutoScaleMode property instead.
System.Windows.Forms. Form	GetAutoS caleSize	This method has been deprecated. Use the ContainerCont rol.AutoScaleDimensions property instead.
System.Windows.Forms. Label	RenderTra nsparent	This property has been deprecated. Use BackColor instea d.
System.Windows.Forms. ListBox	AddItems Core	This method has been deprecated. There is no replacemen t.
System.Windows.Forms. PrintPreviewDialog	AutoSize BaseSize	This property has been deprecated. Use the ContainerCon trol.AutoScaleDimensions property instead.

System.Xaml.dll

Type	Member	Message
System.Windo ws.Markup.Ma rkupExtension ReturnTypeAtt ribute	Expression Type	This isn't used by the XAML parser. See System.Windows.Markup.X amlSetMarkupExtensionAttribute .
System.Windo ws.Markup.Ma rkupExtension ReturnTypeAtt ribute	MarkupExt ensionRetu rnTypeAttri bute(Type, Type)	The <code>expressionType</code> argument isn't used by the XAML parser. To sp ecify the expected return type, use MarkupExtensionReturnTypeAtt ribute(Type) . To specify custom handling for expression types, use S ystem.Windows.Markup.XamlSetMarkupExtensionAttribute .

System.Xml.dll

Type	Member	Message
System.X ml.Validati onType	Auto	Validation type should be specified as ValidationType.DT D or ValidationType.Schema .

Type	Member	Message
System.Xml.XmlValidationType	XDR	XDR validation through System.Xml.XmlValidatingReader is obsolete.
System.Xml.XmlConvert	ToDateTime(String)	Use XmlConvert.ToDateTime(String, XmlDateTimeSerializationMode) .
System.Xml.XmlConvert	ToString(DateTime)	Use XmlConvert.ToString(DateTime, XmlDateTimeSerializationMode) .
System.Xml.XmlReaderSettings	ProhibitDtd	Use the XmlReaderSettings.DtdProcessing property instead.
System.Xml.XmlReaderSettings	XmlReaderSettings(XmlResolver)	First deprecated in the .NET Framework 4.5. Use of this member generates a compiler error. This API supports the .NET Framework infrastructure and isn't intended to be used directly from your code.
System.Xml.XmlTextReader	ProhibitDtd	Use the XmlTextReader.DtdProcessing property instead.
System.Xml.Schema.XmlSchema	Compile(ValidationEventHandler)	Use XmlSchemaSet for schema compilation and validation.
System.Xml.Schema.XmlSchema	Compile(ValidationEventHandler, XmlResolver)	Use XmlSchemaSet for schema compilation and validation.
System.Xml.Schema.XmlSchemaAttribute	AttributeType	This property has been deprecated. Use XmlAttribute.AttributeSchemaType property, which returns a strongly typed attribute type.
System.Xml.Schema.XmlSchemaElement	ElementType	This property has been deprecated. Use the XmlSchemaElement.ElementSchemaType property, which returns a strongly typed element type.

Type	Member	Message
System.Xml.Schema.XmlSchemaType	BaseSchemaType	This property has been deprecated. Use the XmlSchemaType.BaseXmlSchemaType property, which returns a strongly typed base schema type.
System.Xml.Serialization.CodeIdentifier	CodeIdentifier	This class should never get constructed as it contains only static methods.
System.Xml.Serialization.XmlSerializer	FromMappings(XmlMapping[], Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of XmlSerializer.FromMappings that doesn't take an System.Security.Policy.Evidence parameter.
System.Xml.Serialization.XmlSerializer	XmlAttributeOverrides, Type[], XmlRootAttribute, String, String, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an XmlAttributeOverrides constructor overload that doesn't take an System.Security.Policy.Evidence parameter.
System.Xml.Serialization.XmlSerializerFactory	CreateSerializer(Type, XmlAttributeOverrides, Type[], XmlRootAttribute, String, String, Evidence)	This method is obsolete and will be removed in a future release of the .NET Framework. Use an overload of XmlSerializerFactory.CreateSerializer that doesn't take an System.Security.Policy.Evidence parameter.

IEHost.dll and IEExec.exe

The IEHost.dll and IEExec.exe assemblies have been removed from .NET Framework. All of their types and their members are obsolete and are not supported in .NET Framework 4.5 and later. These assemblies were used to host Windows Forms controls and to run executables in Internet Explorer. Alternatives to this technology include ClickOnce, XAML Browser Applications (XBAP), and Microsoft Silverlight.

ISymWrapper.dll

Type	Member	Message
System.Diagnostics.SymbolStore.SymBinder	GetReader(Int32, String, String, String)	The recommended alternative is SymBinder.GetReader(IntPtr, String, String) . ISymbolBinder1.GetReader takes the importer interface pointer as a System.IntPtr instead of an System.Int32 , and thus works on both 32-bit and 64-bit architectures.

Microsoft.Build.Conversion.v4.0.dll

Type	Member	Message
Microsoft.Build.Conversion.ProjectFileConverter	Convert(ProjectLoadSettings)	Use parameterless Convert() overload instead.
Microsoft.Build.Conversion.ProjectFileConverter	Convert(String)	Use parameterless Convert() overload instead.
Microsoft.Build.Conversion.ProjectFileConverter	ConvertInMemory(Engine)	Use parameterless ProjectFileConverter.ConvertInMemory() method instead.
Microsoft.Build.Conversion.ProjectFileConverter	ConvertInMemory(Engine, ProjectLoadSettings)	Use parameterless ProjectFileConverter.ConvertInMemory() method instead.

Microsoft.Build.Engine.dll

Type	Member	Message
Microsoft.Build.Engine	BinPath	Avoid setting BinPath . If you were simply passing in the .NET Framework location as the BinPath , no other action is necessary. Otherwise, define Toolsets instead in the registry or config file, or by adding elements to the Engine's Microsoft.Build.BuildEngine.ToolsetCollection , to use a custom BinPath .
Microsoft.Build.Engine	Engine(String)	If you were simply passing in the .NET Framework location as the BinPath , just change to the parameterless Engine() constructor. Otherwise, you can define custom toolsets in the registry or configuration file, or add elements to the Engine's Microsoft.Build.BuildEngine.ToolsetCollection . Then use either the Engine() or Engine(ToolsetDefinitionLocations) constructor instead.

Microsoft.Build.Framework.dll

Type	Member	Message
------	--------	---------

Type	Member	Message
Microsoft.Build.Framework.XamlTypes.ContentType	ItemGroupName ame	<p>First deprecated in the .NET Framework 4.5.</p> <p>Use of this member generates a compiler error.</p> <p>Use the ContentType.ItemType property instead.</p>

Microsoft.Build.Utilities.v4.0.dll

Type	Member	Message
Microsoft.Build.Utilities.ToolTask	EnvironmentOverride	Use the ToolTask.EnvironmentVariables property.

Microsoft.Data.Entity.Build.Tasks.dll

Type	Member	Message
Microsoft.Data.Entity.Build.Tasks.EntityDeploy	EntityDataModelEmbedde dResources	<p>First deprecated in the .NET Framework 4.5.</p> <p>Used only for version 3.5 backward compatibility.</p>

Microsoft.VisualBasic.dll

Type	Member	Message
Microsoft.VisualBasic.FileSystem	FilePutObject, Obj ect, Object, Obj ect)	This member has been deprecated. Use FileSystem.FilePutObject to write Object types, or coerce FileNumber and RecordNumber to Int32 for writing non-object types.
Microsoft.VisualBasic.CompilerServices.CompilerServices	FallbackUserDefi nedConversion	<p>Use of this member generates a compiler error.</p> <p>Don't use this method.</p>

Type	Member	Message
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackC all	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	Fallback Get	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackI ndexSet	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackI ndexSet Complex	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackI nvokede fault1	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackI nvokede fault2	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackS et	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.NewLa teBinding	FallbackS etCompl ex	Use of this member generates a compiler error. Don't use this method.
Microsoft.Visua lBasic.Compiler Services.Opera tors	FallbackI nvokeUs erDefine dOperat or	Use of this member generates a compiler error. Don't use this method.

Type	Member	Message
Microsoft.Visua lBasic.MyServic es.RegistryProx y	DynData	The <code>DynData</code> registry key works only on Win9x, which isn't supported by this version of .NET Framework. Use the <code>PerformanceData</code> registry key instead. This property will be removed from a future version of .NET Framework.

See also

- [What's obsolete in .NET Framework](#)
- [Obsolete types](#)

Code analysis

Article • 11/17/2022

You can use code analyzers to find potential issues in your .NET Framework application code. The analyzers find potential issues and suggest fixes for them.

Roslyn-based code analyzers run interactively in Visual Studio as you write your code or as part of a CI build. You should add the analyzers to your project as early as possible in the development cycle. The sooner you find any potential issues in your code, the easier they are to fix. The analyzers flag issues in existing code and warn about new issues as you continue development.

ⓘ Note

This article makes use of the now deprecated [Microsoft.NetFramework.Analyzers NuGet package](#). Starting in .NET 5, NET analyzers are included with the .NET SDK. If needed, you can use the [Microsoft.CodeAnalysis.NetAnalyzers NuGet package](#) instead. For more information, see [Code analysis in .NET](#).

Install and configure analyzers

The .NET Framework Analyzer is delivered in the [Microsoft.NetFramework.Analyzers NuGet package](#). This package provides analyzers that are specific to .NET Framework APIs, which includes security analyzers. The package is included with the [Microsoft.CodeAnalysis.FxCopAnalyzers package](#), so if you install that package, there's no need to install the .NET Framework analyzers separately.

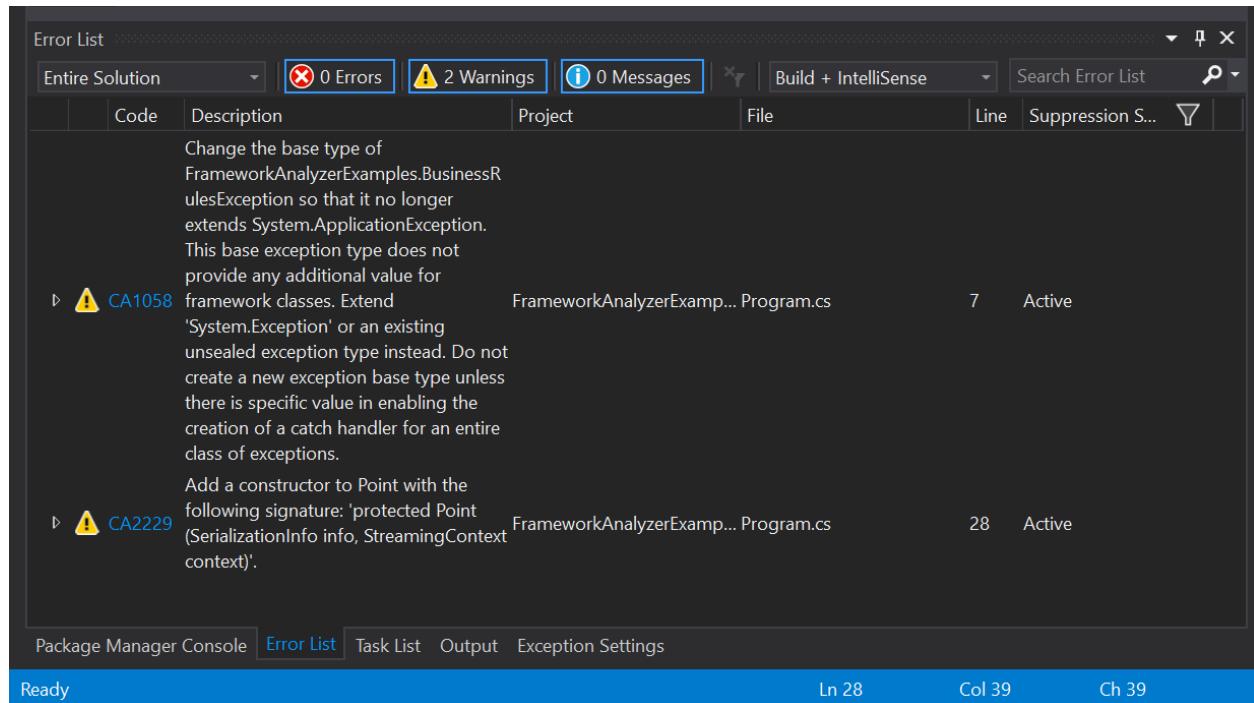
Install the NuGet package on every project where you want the analyzers to run. Only one developer needs to add them to the project. The analyzer package is a project dependency and will run on every developer's machine once it has the updated solution.

To install the package, right-click on the project, and select "Manage Dependencies". From the NuGet explorer, search for "Microsoft.NetFramework.Analyzers". Install the latest stable version in all projects in your solution.

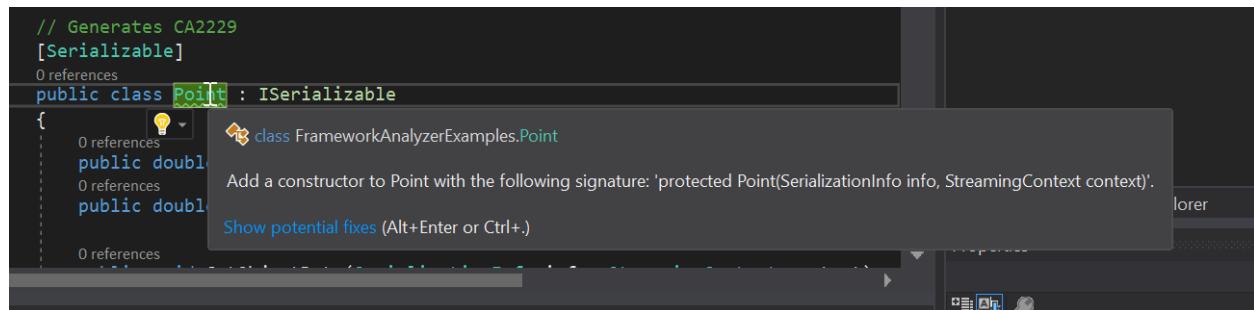
Use the analyzers

Once the NuGet package is installed, build your solution. The analyzer will report any issues it locates in your codebase. The issues are reported as warnings in the Visual

Studio Error List window, as shown in the following image:



As you write code, you see squiggles underneath any potential issue in your code. Hover over any issue to get more information and see suggestions for any possible fix, as shown in the following image:



For more information, see [Code analysis in Visual Studio](#).

Types of rules

The analyzers examine the code in your solution and surface warnings with a CA prefix. For a list of all possible warnings, see [Code quality rules](#). Only some of these warnings apply to .NET Framework APIs, including:

- [CA1058: Types should not extend certain base types](#)
- [CA2153: Do not catch corrupted state exceptions](#)
- [CA2229: Implement serialization constructors](#)
- [CA2235: Mark all non-serializable fields](#)

- CA2237: Mark `ISerializable` types with `SerializableAttribute`
- CA3075: Insecure DTD processing in XML
- CA5350: Do not use weak cryptographic algorithms
- CA5351 Do not use broken cryptographic algorithms

See also

- [Code analysis in Visual Studio](#)
- [Code analysis in the .NET SDK](#)