**.NET architectural components**

* 10/05/2020

A .NET app is developed for and runs in one or more *implementations of .NET*. Implementations of .NET include the .NET Framework, .NET 5 (and .NET Core), and Mono. There is an API specification common to multiple implementations of .NET that's called .NET Standard. This article gives a brief introduction to each of these concepts.

**.NET Standard**

.NET Standard is a set of APIs that are implemented by the Base Class Library of a .NET implementation. More formally, it's a specification of .NET APIs that make up a uniform set of contracts that you compile your code against. These contracts are implemented in multiple .NET implementations.

.NET Standard is a [target framework](https://docs.microsoft.com/en-us/dotnet/standard/glossary#target-framework). If your code targets a version of .NET Standard, it can run on any .NET implementation that supports that version of .NET Standard.

.NET Standard was created to enable portability across different .NET implementations, but now .NET 5 offers a better way to share code across multiple platforms and workloads. For more information, see [.NET 5 and .NET Standard](https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-5-and-net-standard).

**.NET implementations**

Each implementation of .NET includes the following components:

* One or more runtimes. Examples: .NET Framework CLR, .NET 5 CLR.
* A class library. Examples: .NET Framework Base Class Library, .NET 5 Base Class Library.
* Optionally, one or more application frameworks. Examples: [ASP.NET](https://www.asp.net/), [Windows Forms](https://docs.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview), and [Windows Presentation Foundation (WPF)](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/) are included in the .NET Framework and .NET 5.
* Optionally, development tools. Some development tools are shared among multiple implementations.

There are four .NET implementations that Microsoft supports:

* .NET 5 (and .NET Core) and later versions
* .NET Framework
* Mono
* UWP

.NET 5 is now the primary implementation, the one that is the focus of ongoing development. .NET 5 is built on a single code base that supports multiple platforms and many workloads, such as Windows desktop apps and cross-platform console apps, cloud services, and websites.

**.NET 5**

.NET 5 is a cross-platform implementation of .NET that is designed to handle server and cloud workloads at scale. It also supports other workloads, including desktop apps. It runs on Windows, macOS, and Linux. It implements .NET Standard, so code that targets .NET Standard can run on .NET 5. [ASP.NET Core](https://dotnet.microsoft.com/learn/aspnet/what-is-aspnet-core), [Windows Forms](https://docs.microsoft.com/en-us/dotnet/desktop/winforms/windows-forms-overview), and [Windows Presentation Foundation (WPF)](https://docs.microsoft.com/en-us/dotnet/desktop/wpf/) all run on .NET 5.

For more information, see the following resources:

* [.NET introduction](https://docs.microsoft.com/en-us/dotnet/core/introduction)
* [Choosing between .NET 5 and .NET Framework for server apps](https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server)
* [.NET 5 and .NET Standard](https://docs.microsoft.com/en-us/dotnet/standard/net-standard#net-5-and-net-standard)

**.NET Framework**

.NET Framework is the original .NET implementation that has existed since 2002. Versions 4.5 and later implement .NET Standard, so code that targets .NET Standard can run on those versions of .NET Framework. It contains additional Windows-specific APIs, such as APIs for Windows desktop development with Windows Forms and WPF. .NET Framework is optimized for building Windows desktop applications.

For more information, see the [.NET Framework guide](https://docs.microsoft.com/en-us/dotnet/framework/).

**Mono**

Mono is a .NET implementation that is mainly used when a small runtime is required. It is the runtime that powers Xamarin applications on Android, macOS, iOS, tvOS, and watchOS and is focused primarily on a small footprint. Mono also powers games built using the Unity engine.

It supports all of the currently published .NET Standard versions.

Historically, Mono implemented the larger API of the .NET Framework and emulated some of the most popular capabilities on Unix. It is sometimes used to run .NET applications that rely on those capabilities on Unix.

Mono is typically used with a just-in-time compiler, but it also features a full static compiler (ahead-of-time compilation) that is used on platforms like iOS.

For more information, see the [Mono documentation](https://www.mono-project.com/docs/).

**Universal Windows Platform (UWP)**

UWP is an implementation of .NET that is used for building modern, touch-enabled Windows applications and software for the Internet of Things (IoT). It's designed to unify the different types of devices that you may want to target, including PCs, tablets, phones, and even the Xbox. UWP provides many services, such as a centralized app store, an execution environment (AppContainer), and a set of Windows APIs to use instead of Win32 (WinRT). Apps can be written in C++, C#, Visual Basic, and JavaScript.

For more information, see [Introduction to the Universal Windows Platform](https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide).

**.NET runtimes**

A runtime is the execution environment for a managed program. The OS is part of the runtime environment but is not part of the .NET runtime. Here are some examples of .NET runtimes:

* Common Language Runtime (CLR) for .NET Framework
* Common Language Runtime (CLR) for .NET 5
* .NET Native for Universal Windows Platform
* The Mono runtime for Xamarin.iOS, Xamarin.Android, Xamarin.Mac, and the Mono desktop framework

**.NET tooling and common infrastructure**

You have access to an extensive set of tools and infrastructure components that work with every implementation of .NET. These tools and components include:

* The .NET languages and their compilers
* The .NET project system (based on *.csproj*, *.vbproj*, and *.fsproj* files)
* [MSBuild](https://docs.microsoft.com/en-us/visualstudio/msbuild/msbuild), the build engine used to build projects
* [NuGet](https://docs.microsoft.com/en-us/nuget/), Microsoft's package manager for .NET
* Open-source build orchestration tools, such as [CAKE](https://cakebuild.net/) and [FAKE](https://fake.build/)

For more information, see [Tools and productivity](https://docs.microsoft.com/en-us/dotnet/core/introduction#tools-and-productivity).

**Applicable standards**

The C# Language and the Common Language Infrastructure (CLI) specifications are standardized through [Ecma International®](https://www.ecma-international.org/). The first editions of these standards were published by Ecma in December 2001.

Subsequent revisions to the standards have been developed by the TC49-TG2 (C#) and TC49-TG3 (CLI) task groups within the Programming Languages Technical Committee ([TC49](https://www.ecma-international.org/memento/tc49.htm)), and adopted by the Ecma General Assembly and subsequently by ISO/IEC JTC 1 via the ISO Fast-Track process.

**Latest standards**

The following official Ecma documents are available for [C#](http://www.ecma-international.org/publications/standards/Ecma-334.htm) and the [CLI](http://www.ecma-international.org/publications/standards/Ecma-335.htm) ([TR-84](http://www.ecma-international.org/publications/techreports/E-TR-084.htm)):

* **The C# Language Standard (version 5.0)**: [ECMA-334.pdf](https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-334.pdf)
* **The Common Language Infrastructure**: Available in [pdf](https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf) form and [zip](https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.zip) form.
* **Information Derived from the Partition IV XML File**: Available in [pdf](https://www.ecma-international.org/publications/files/ECMA-TR/ECMA%20TR-084.pdf) and [zip](https://www.ecma-international.org/publications/files/ECMA-TR/TR-084.zip) formats.

The official ISO/IEC documents are available from the ISO/IEC [Publicly Available Standards](https://standards.iso.org/ittf/PubliclyAvailableStandards/) page. These links are direct from that page:

* **Information technology - Programming languages - C#**: [ISO/IEC 23270:2018](https://standards.iso.org/ittf/PubliclyAvailableStandards/c075178_ISO_IEC_23270_2018.zip)
* **Information technology — Common Language Infrastructure (CLI) Partitions I to VI**: [ISO/IEC 23271:2012](https://standards.iso.org/ittf/PubliclyAvailableStandards/c058046_ISO_IEC_23271_2012%28E%29.zip)
* **Information technology — Common Language Infrastructure (CLI) — Technical Report on Information Derived from Partition IV XML File**: [ISO/IEC TR 23272:2011](https://standards.iso.org/ittf/PubliclyAvailableStandards/c057955_ISO_IEC_TR_23272_2011.zip)