**Microsoft Releases .NET Framework 4.7.2**

Microsoft's new .NET Core 2.0 and .NET Standard 2.0 offerings may be generating the most buzz among .NET developers these days, but for many use cases the traditional .NET Framework is still the best choice, just released in version 4.7.2.

The new [**.NET Framework 4.7.2**](https://docs.microsoft.com/en-us/dotnet/framework/whats-new/#v472) is the next major update following the October 2017 [**release of v4.7.1**](https://visualstudiomagazine.com/articles/2017/10/18/net-framework-update.aspx), which added support for .NET Standard 2.0, defining APIs that all conformant .NET implementations must provide to ensure consistent API usage across development projects, replacing the previous Portable Class Libraries (PCL) as the means to create libraries for all scenarios.

While .NET Core offers cross-platform functionality and more, the 16-year-old .NET Framework is still an optimal choice for targeting Windows desktop projects such as WinForms, WPF and ASP.NET WebForms apps.

Both .NET Core and .NET Framework are used for creating server-side apps, and Microsoft [**says**](https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server) in determining which to use, to consider:

* Using .NET Framework for your server application when:
  + Your app currently uses .NET Framework (recommendation is to extend instead of migrating).
  + Your app uses third-party .NET libraries or NuGet packages not available for .NET Core.
  + Your app uses .NET technologies that aren't available for .NET Core.
  + Your app uses a platform that doesn’t support .NET Core.
* Using .NET Core for your server application when:
  + You have cross-platform needs.
  + You are targeting microservices.
  + You are using Docker containers.
  + You need high-performance and scalable systems.
  + You need side-by-side .NET versions per application.

Microsoft also provided [**guidance**](http://visualstudiomagazine.com/articles/2017/11/29/porting-to-net-core.aspx) about when and when not to consider porting existing .NET Framework projects to .NET Core.

As noted one of main use cases for .NET Framework is ASP.NET Web apps, and in that area the new 4.7.2 release has added several improvements, Microsoft said in a [**post**](https://blogs.msdn.microsoft.com/dotnet/2018/04/30/announcing-the-net-framework-4-7-2/) yesterday (April 30), notably:

* Support for dependency injection (DI) in Web Forms apps. While DI is already supported by the [**MVC Framework**](https://msdn.microsoft.com/library/dd381412(v=vs.108).aspx), it was much harder to implement for ASP.NET Web Forms. Microsoft describes DI as "a technique whereby one object supplies the dependencies of another object. It decouples the objects so that no client code has to be changed simply because an object it depends on needs to be changed to a different one." Microsoft said the new DI functionality in ASP.NET Web Forms applications will support:
  + Setter-based, interface-based and constructor-based injection in Web application projects in Handler, Module, Page, User control and Custom control.
  + Setter-based and interface-based injection in Web site projects in Handler, Module, Page, User controls and Custom controls.
  + Extensibility to support different DI frameworks.

* [**SameSite Cookie**](https://tools.ietf.org/html/draft-west-first-party-cookies-07). This prevents browsers from sending this cookie along with cross-site requests, Microsoft said. "In .NET Framework 4.7.2, a new property **[SameSite](https://docs.microsoft.com/en-us/dotnet/api/system.web.httpcookie.samesite?view=netframework-4.7.2" \l "System_Web_HttpCookie_SameSite" \t "_blank)** has been added in **[HttpCookie](https://docs.microsoft.com/dotnet/api/system.web.httpcookie?view=netframework-4.7.2" \t "_blank)** type and ASP.NET will add a SameSite attribute into the set-cookie header if HttpCookie.SameSite is set to SameSiteMode.Strict or SameSiteMode.Lax. The support for SameSite cookie is two-fold in this case:
  + In HttpCookie object
  + In FormsAuthentication and SessionState cookies

Microsoft has also beefed up SQL query security with the addition of a new Azure AD Interactive authentication keyword to support [**Multi-Factor Authentication**](https://docs.microsoft.com/azure/sql-database/sql-database-ssms-mfa-authentication) (MFA). That will prevent the need to include a password directly in a connection string. It also enables support for [**Azure AD Authentication**](https://docs.microsoft.com/azure/sql-database/sql-database-aad-authentication-configure).

A host of other improvements, bug fixes and tweaks were made to the various .NET Framework components, and you can check them out in the [**.NET Framework 4.7.2 Release Notes**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/README.md) on GitHub. Specific sections include:

* [**ASP.NET**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#aspnet)
* [**Base Class Library (BCL)**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#bcl)
* [**Common Language Runtime (CLR)**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#clr)
* [**ClickOnce**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#clickonce)
* [**Networking**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#networking)
* [**SQL**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#sql)
* [**Windows Communication Foundation (WCF)**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#wcf)
* [**Windows Forms**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#windows-forms)
* [**Workflow**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#workflow)
* [**Windows Presentation Foundation (WPF)**](https://github.com/Microsoft/dotnet/blob/master/releases/net472/dotnet472-changes.md#wpf)

.NET Framework 4.7.2 is included in the [**Windows 10 April 2018 Update**](https://blogs.windows.com/windowsexperience/2018/04/30/how-to-get-the-windows-10-april-2018-update/) and can be installed for Windows 7+ and Windows Server 2008 R2+ via a [**Web installer**](https://go.microsoft.com/fwlink/?LinkId=863262) or [**offline installer**](https://go.microsoft.com/fwlink/?LinkId=863265) (note that clicking on links initiates a download).

Microsoft yesterday [**announced**](https://blogs.windows.com/buildingapps/2018/04/30/start-developing-on-windows-10-april-2018-update-today/#FuQfUR30oyV88lYQ.97) that the Windows 10 April 2018 Update (build 17134.1) and SDK are now available.