

Target frameworks

📅 05/31/2018 ⌚ 4 minutes to read Contributors       all

In this article

[Latest target framework versions](#)

[Supported target framework versions](#)

[How to specify target frameworks](#)

[Deprecated target frameworks](#)

[See also](#)

When you target a framework in an app or library, you're specifying the set of APIs that you'd like to make available to the app or library. You specify the target framework in your project file using Target Framework Monikers (TFMs).

An app or library can target a version of [.NET Standard](#). .NET Standard versions represent standardized sets of APIs across all .NET implementations. For example, a library can target .NET Standard 1.6 and gain access to APIs that function across .NET Core and .NET Framework using the same codebase.

An app or library can also target a specific .NET implementation to gain access to implementation-specific APIs. For example, an app that targets Xamarin.iOS (for example, `Xamarin.iOS10`) gets access to Xamarin-provided iOS API wrappers for iOS 10, or an app that targets the Universal Windows Platform (UWP, `uap10.0`) has access to APIs that compile for devices that run Windows 10.

For some target frameworks (for example, the .NET Framework), the APIs are defined by the assemblies that the framework installs on a system and may include application framework APIs (for example, ASP.NET).

For package-based target frameworks (for example, .NET Standard and .NET Core), the APIs are defined by the packages included in the app or library. A *metapackage* is a NuGet package that has no content of its own but is a list of dependencies (other packages). A NuGet package-based target framework implicitly specifies a metapackage that references all the packages that together make up the framework.

Latest target framework versions

The following table defines the most common target frameworks, how they're referenced, and which version of the [.NET Standard](#) they implement. These target framework versions are the latest stable versions. Pre-release versions aren't shown. A Target Framework Moniker (TFM) is a standardized token format for specifying the target framework of a .NET app or library.

Target Framework	Latest Stable Version	Target Framework Moniker (TFM)	Implemented .NET Standard Version
.NET Standard	2.0	netstandard2.0	N/A
.NET Core	2.1	netcoreapp2.1	2.0
.NET Framework	4.7.2	net472	2.0

Supported target framework versions


A target framework is typically referenced by a TFM. The following table shows the target frameworks supported by the .NET Core SDK and the NuGet client. Equivalents are shown within brackets. For example, `win81` is an equivalent TFM to `netcore451`.

Target Framework	TFM
.NET Standard	netstandard1.0
	netstandard1.1
	netstandard1.2
	netstandard1.3
	netstandard1.4
	netstandard1.5
	netstandard1.6
.NET Core	netstandard2.0
	netcoreapp1.0
	netcoreapp1.1
	netcoreapp2.0
	netcoreapp2.1
.NET Framework	net11
	net20
	net35
	net40
	net403
	net45
	net451
	net452
	net46
	net461
	net462
	net47
	net471
	net472

Target Framework	TFM
Windows Store	netcore [netcore45] netcore45 [win] [win8] netcore451 [win81]
.NET Micro Framework	netmf
Silverlight	sl4 sl5
Windows Phone	wp [wp7] wp7 wp75 wp8 wp81 wpa81
Universal Windows Platform	uap [uap10.0] uap10.0 [win10] [netcore50]

How to specify target frameworks

Target frameworks are specified in your project file. When a single target framework is specified, use the **TargetFramework** element. The following console app project file demonstrates how to target .NET Core 2.0:

XML	 Copy
<pre><Project Sdk="Microsoft.NET.Sdk"> <PropertyGroup> <OutputType>Exe</OutputType> <TargetFramework>netcoreapp2.0</TargetFramework> </PropertyGroup> </Project></pre>	

When you specify multiple target frameworks, you may conditionally reference assemblies for each target framework. In your code, you can conditionally compile against those assemblies by using preprocessor symbols with *if-then-else* logic.

The following library project file targets APIs of .NET Standard (`netstandard1.4`) and APIs of the .NET Framework (`net40` and `net45`). Use the plural **TargetFrameworks** element with multiple

target frameworks. Note how the `Condition` attributes include implementation-specific packages when the library is compiled for the two .NET Framework TFMs:

XML

Copy

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <TargetFrameworks>netstandard1.4;net40;net45</TargetFrameworks>
  </PropertyGroup>

  <!-- Conditionally obtain references for the .NET Framework 4.0 target -->
  <ItemGroup Condition=" '$(TargetFramework)' == 'net40' ">
    <Reference Include="System.Net" />
  </ItemGroup>

  <!-- Conditionally obtain references for the .NET Framework 4.5 target -->
  <ItemGroup Condition=" '$(TargetFramework)' == 'net45' ">
    <Reference Include="System.Net.Http" />
    <Reference Include="System.Threading.Tasks" />
  </ItemGroup>

</Project>
```

Within your library or app, you write conditional code to compile for each target framework:

C#

Copy

```
public class MyClass
{
    static void Main()
    {
#if NET40
        Console.WriteLine("Target framework: .NET Framework 4.0");
#elif NET45
        Console.WriteLine("Target framework: .NET Framework 4.5");
#else
        Console.WriteLine("Target framework: .NET Standard 1.4");
#endif
    }
}
```

The build system is aware of preprocessor symbols representing the target frameworks shown in the [Supported target framework versions](#) table. When using a symbol that represents a .NET Standard or .NET Core TFM, replace the dot with an underscore and change lowercase letters to uppercase (for example, the symbol for `netstandard1.4` is `NETSTANDARD1_4`).

The complete list of preprocessor symbols for .NET Core target frameworks is:

Target Frameworks	Symbols
.NET Framework	NET20 , NET35 , NET40 , NET45 , NET451 , NET452 , NET46 , NET461 , NET462 , NET47 , NET471 , NET472
.NET Standard	NETSTANDARD1_0 , NETSTANDARD1_1 , NETSTANDARD1_2 , NETSTANDARD1_3 , NETSTANDARD1_4 , NETSTANDARD1_5 , NETSTANDARD1_6 , NETSTANDARD2_0
.NET Core	NETCOREAPP1_0 , NETCOREAPP1_1 , NETCOREAPP2_0 , NETCOREAPP2_1

Deprecated target frameworks

The following target frameworks are deprecated. Packages targeting these target frameworks should migrate to the indicated replacements.

Deprecated TFM	Replacement
aspnet50 aspnetcore50 dnxcore50 dnx dnx45 dnx451 dnx452	netcoreapp
dotnet dotnet50 dotnet51 dotnet52 dotnet53 dotnet54 dotnet55 dotnet56	netstandard
netcore50	uap10.0
win	netcore45
win8	netcore45
win81	netcore451

Deprecated TFM	Replacement
win10	uap10.0
winrt	netcore45

See also

[Packages, Metapackages and Frameworks](#)
[Developing Libraries with Cross Platform Tools](#)
[.NET Standard](#)
[.NET Core Versioning](#)
[dotnet/standard GitHub repository](#)
[NuGet Tools GitHub Repository](#)
[Framework Profiles in .NET](#)