

How to: Determine which .NET Framework versions are installed

04/10/2018 12 minutes to read Contributors      all

In this article

[To find .NET Framework versions by viewing the registry \(.NET Framework 1-4\)](#)

[To find .NET Framework versions by viewing the registry \(.NET Framework 4.5 and later\)](#)

[To find .NET Framework versions by querying the registry in code \(.NET Framework 1-4\)](#)

[To find .NET Framework versions by querying the registry in code \(.NET Framework 4.5 and later\)](#)

[To check for a minimum-required .NET Framework version by querying the registry in PowerShell \(.NET Framework 4.5 and later\)](#)

[To find the current runtime version by using the Clrver tool](#)

[To find the current runtime version by querying the Environment class in code](#)

[See also](#)

Users can install and run multiple versions of the .NET Framework on their computers. When you develop or deploy your app, you might need to know which .NET Framework versions are installed on the user's computer. Note that the .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. The .NET Framework and assemblies share the same version number.
- The common language runtime (CLR), which manages and executes your app's code. The CLR is identified by its own version number (see [Versions and Dependencies](#)).

To get an accurate list of the .NET Framework versions installed on a computer, you can view the registry or query the registry in code:

[Viewing the registry \(versions 1-4\)](#)

[Viewing the registry \(version 4.5 and later\)](#)

[Using code to query the registry \(versions 1-4\)](#)

[Using code to query the registry \(version 4.5 and later\)](#)

[Using PowerShell to query the registry \(version 4.5 and later\)](#)

To find the CLR version, you can use a tool or code:

[Using the Clrver tool](#)

[Using code to query the System.Environment class](#)

For information about detecting the installed updates for each version of the .NET Framework, see [How to: Determine Which .NET Framework Updates Are Installed](#). For information about installing the .NET Framework, see [Install the .NET Framework for developers](#).

To find .NET Framework versions by viewing the registry (.NET Framework 1-4)

1. On the **Start** menu, choose **Run**.
2. In the **Open** box, enter **regedit.exe**.

You must have administrative credentials to run regedit.exe.

3. In the Registry Editor, open the following subkey:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP
```

The installed versions are listed under the NDP subkey. The version number is stored in the **Version** entry. For the .NET Framework 4 the **Version** entry is under the Client or Full subkey (under NDP), or under both subkeys.

ⓘ Note

The "NET Framework Setup" folder in the registry does not begin with a period.

To find .NET Framework versions by viewing the registry (.NET Framework 4.5 and later)

1. On the **Start** menu, choose **Run**.
2. In the **Open** box, enter **regedit.exe**.

You must have administrative credentials to run regedit.exe.

3. In the Registry Editor, open the following subkey:

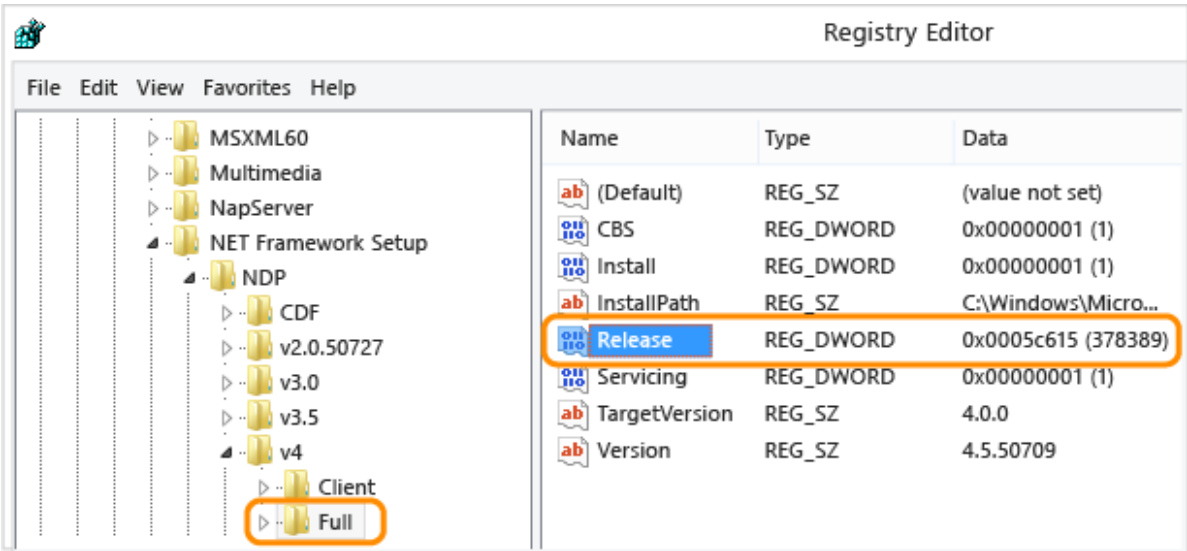
```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full
```

Note that the path to the **Full** subkey includes the subkey **Net Framework** rather than **.NET Framework**.

ⓘ Note

If the **Full** subkey is not present, then you do not have the .NET Framework 4.5 or later installed.

Check for a DWORD value named **Release**. The existence of the **Release** DWORD indicates that the .NET Framework 4.5 or newer has been installed on that computer.



The value of the `Release` DWORD indicates which version of the .NET Framework is installed.

Important

The following table lists the keys of released versions of the .NET Framework only. It doesn't list the keys of preview or pre-release versions.

Value of the Release DWORD	Version
378389	.NET Framework 4.5
378675	.NET Framework 4.5.1 installed with Windows 8.1 or Windows Server 2012 R2
378758	.NET Framework 4.5.1 installed on Windows 8, Windows 7 SP1, or Windows Vista SP2
379893	.NET Framework 4.5.2
On Windows 10 systems only: 393295	.NET Framework 4.6
On all other OS versions: 393297	
On Windows 10 November Update systems only: 394254	.NET Framework 4.6.1
On all other OS versions: 394271	
On Windows 10 Anniversary Update and Windows Server 2016: 394802	.NET Framework 4.6.2
On all other OS versions: 394806	

Value of the Release DWORD	Version
On Windows 10 Creators Update only: 460798	.NET Framework 4.7
On all other OS versions: 460805	
On Windows 10 Fall Creators Update only: 461308	.NET Framework 4.7.1
On all other OS versions: 461310	
On Windows 10 April 2018 Update only: 461808	.NET Framework 4.7.2
On all other OS versions, including Windows 10 October 2018 Update: 461814	

To find .NET Framework versions by querying the registry in code (.NET Framework 1-4)

- Use the [Microsoft.Win32.RegistryKey](#) class to access the Software\Microsoft\NET Framework Setup\NDP\ subkey under HKEY_LOCAL_MACHINE in the Windows registry.

The following code shows an example of this query.

ⓘ Note

This code does not show how to detect the .NET Framework 4.5 or later. Check the [Release](#) DWORD to detect those versions, as described in the previous section. For code that does detect the .NET Framework 4.5 or later versions, see the next section in this article.

C#

 Copy

```
using Microsoft.Win32;
using System;

public static class VersionTest
{
    public static void Main()
    {
        GetVersionFromRegistry();
    }

    private static void GetVersionFromRegistry()
    {
        // Opens the registry key for the .NET Framework entry.
        using (RegistryKey ndpKey =
            RegistryKey.OpenRemoteBaseKey(RegistryHive.LocalMachine, "").
```

```

OpenSubKey(@"SOFTWARE\Microsoft\NET Framework Setup\NDP\"))
{
    // As an alternative, if you know the computers you will query are
running .NET Framework 4.5
    // or later, you can use:
    // using (RegistryKey ndpKey =
RegistryKey.OpenBaseKey(RegistryHive.LocalMachine,
    // RegistryView.Registry32).OpenSubKey(@"SOFTWARE\Microsoft\NET
Framework Setup\NDP\"))
    foreach (string versionKeyName in ndpKey.GetSubKeyNames())
    {
        if (versionKeyName.StartsWith("v"))
        {
            RegistryKey versionKey = ndpKey.OpenSubKey(versionKeyName);
            string name = (string)versionKey.GetValue("Version", "");
            string sp = versionKey.GetValue("SP", "").ToString();
            string install = versionKey.GetValue("Install", "").ToString();
            if (install == "") //no install info, must be later.
                Console.WriteLine(versionKeyName + " " + name);
            else
            {
                if (sp != "" && install == "1")
                {
                    Console.WriteLine(versionKeyName + " " + name + " SP" +
sp);
                }
            }
            if (name != "")
            {
                continue;
            }
            foreach (string subKeyName in versionKey.GetSubKeyNames())
            {
                RegistryKey subKey = versionKey.OpenSubKey(subKeyName);
                name = (string)subKey.GetValue("Version", "");
                if (name != "")
                {
                    sp = subKey.GetValue("SP", "").ToString();
                    install = subKey.GetValue("Install", "").ToString();
                    if (install == "") //no install info, must be later.
                        Console.WriteLine(versionKeyName + " " + name);
                    else
                    {
                        if (sp != "" && install == "1")
                        {
                            Console.WriteLine(" " + subKeyName + " " + name + "
SP" + sp);
                        }
                        else if (install == "1")
                        {
                            Console.WriteLine(" " + subKeyName + " " + name);
                        }
                    }
                }
            }
        }
    }
}

```

```
}
  }
    }
      }
```

The example produces output that's similar to the following:

Copy

```
v2.0.50727 2.0.50727.4016 SP2
v3.0 3.0.30729.4037 SP2
v3.5 3.5.30729.01 SP1
v4
  Client 4.0.30319
  Full 4.0.30319
```

To find .NET Framework versions by querying the registry in code (.NET Framework 4.5 and later)

1. The existence of the `Release` DWORD indicates that the .NET Framework 4.5 or later has been installed on a computer. The value of the keyword indicates the installed version. To check this keyword, use the `OpenBaseKey` and `OpenSubKey` methods of the `Microsoft.Win32.RegistryKey` class to access the `Software\Microsoft\NET Framework Setup\NDP\v4\Full` subkey under `HKEY_LOCAL_MACHINE` in the Windows registry.
2. Check the value of the `Release` keyword to determine the installed version. To be forward-compatible, you can check for a value greater than or equal to the values listed in the table. Here are the .NET Framework versions and associated `Release` keywords.

Important

The following table lists the keys of released versions of the .NET Framework only. It doesn't list the keys of preview or pre-release versions.

Version	Value of the Release DWORD
.NET Framework 4.5	378389
.NET Framework 4.5.1 installed with Windows 8.1	378675
.NET Framework 4.5.1 installed on Windows 8, Windows 7 SP1, or Windows Vista SP2	378758

Version	Value of the Release DWORD
.NET Framework 4.5.2	379893
.NET Framework 4.6 installed with Windows 10	393295
.NET Framework 4.6 installed on all other Windows OS versions	393297
.NET Framework 4.6.1 installed on Windows 10	394254
.NET Framework 4.6.1 installed on all other Windows OS versions	394271
.NET Framework 4.6.2 installed on Windows 10 Anniversary Update and Windows Server 2016	394802
.NET Framework 4.6.2 installed on all other Windows OS versions	394806
.NET Framework 4.7 installed on Windows 10 Creators Update	460798
.NET Framework 4.7 installed on all other Windows OS versions	460805
.NET Framework 4.7.1 installed on Windows 10 Fall Creators Update	461308
.NET Framework 4.7.1 installed on all other Windows OS versions	461310
.NET Framework 4.7.2 installed on Windows 10 October 2018 Update	461814
.NET Framework 4.7.2 installed on Windows 10 April 2018 Update	461808
.NET Framework 4.7.2 installed on Windows 10 Fall Creators Update and earlier OS versions	461814

The following example checks the `Release` value in the registry to determine whether the .NET Framework 4.5 or a later version of the .NET Framework is installed.

C#	 Copy
<pre>using System; using Microsoft.Win32; public class GetDotNetVersion { public static void Main() { GetDotNetVersion.Get45PlusFromRegistry(); } }</pre>	

```

private static void Get45PlusFromRegistry()
{
    const string subkey = @"SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full\";

    using (RegistryKey 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 >= will enable forward compatibility.
private static string CheckFor45PlusVersion(int releaseKey)
{
    if (releaseKey >= 461808)
        return "4.7.2 or later";
    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";
}
}

// This example displays output like the following:
//      .NET Framework Version: 4.6.1

```

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.

To check for a minimum-required .NET Framework version by querying the registry in PowerShell (.NET Framework 4.5 and later)

- The following example checks the value of the `Release` keyword to determine whether .NET Framework 4.6.2 or higher is installed, regardless of Windows OS version (returning `True` if it is and `False` otherwise).

PowerShell 

```
# PowerShell 5
Get-ChildItem 'HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full\' | Get-ItemPropertyValue -Name Release | Foreach-Object { $_ -ge 394802 }
```

PowerShell 

```
# PowerShell 4
(Get-ItemProperty "HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full").Release -gt 394802
```

You can replace `394802` in the previous example with another value from the following table to check for a different minimum-required .NET Framework version.

Version	Minimum value of the Release DWORD
.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

To find the current runtime version by using the Clrver tool

- Use the CLR Version Tool (Clrver.exe) to determine which versions of the common language runtime are installed on a computer.

From a Visual Studio Command Prompt, enter `clrver`. This command produces output similar to the following:

	Copy
Versions installed on the machine: v2.0.50727 v4.0.30319	

For more information about using this tool, see [Clrver.exe \(CLR Version Tool\)](#).

To find the current runtime version by querying the Environment class in code

- Query the [Environment.Version](#) property to retrieve a [Version](#) object that identifies the version of the runtime that is currently executing the code. You can use the [Version.Major](#) property to get the major release identifier (for example, "4" for version 4.0), the [Version.Minor](#) property to get the minor release identifier (for example, "0" for version 4.0), or the [Object.ToString](#) method to get the entire version string (for example, "4.0.30319.18010", as shown in the following code). This property returns a single value that reflects the version of the runtime that is currently executing the code; it does not return assembly versions or other versions of the runtime that may have been installed on the computer.

For the .NET Framework Versions 4, 4.5, 4.5.1, and 4.5.2, the [Environment.Version](#) property returns a [Version](#) object whose string representation has the form `4.0.30319.xxxxx`. For the .NET Framework 4.6 and later, it has the form `4.0.30319.42000`.

❗ Important

For the .NET Framework 4.5 and later, we do not recommend using the [Environment.Version](#) property to detect the version of the runtime. Instead, we recommend that you query the registry, as described in the [To find .NET Framework versions by querying the registry in code \(.NET Framework 4.5 and later\)](#) section earlier in this article.

Here's an example of querying the [Environment.Version](#) property for runtime version information:

C#	Copy
----	------

```
using System;

public class VersionTest
{
    public static void Main()
    {
        Console.WriteLine($"Version: {Environment.Version}");
    }
}
```

The example produces output that's similar to the following:

 Copy

Version: 4.0.30319.18010

See also

[How to: Determine Which .NET Framework Updates Are Installed](#)

[Install the .NET Framework for developers](#)

[Versions and Dependencies](#)