# How to determine the windows version by using Visual C#

## Introduction

This sample is based on the topic: [How to determine the Windows Version by using Visual C#](https://support.microsoft.com/en-us/kb/304283).

This sample is a console application which demonstrates how to determine the windows version by using Visual C#.

## Sample prerequisites

To open and run this sample, ensure that the following requisites have been met:

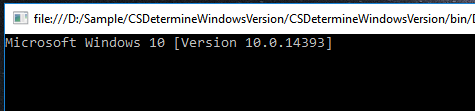
* Microsoft Visual Studio 2015 Update3 or later version(s)

## Building the sample

* Open the sample solution “**CSDetermineWindowsVersion.sln**” using Visual Studio.
* Press **F6 Key** or **select Build -> Build Solution** from the menu to build the sample.

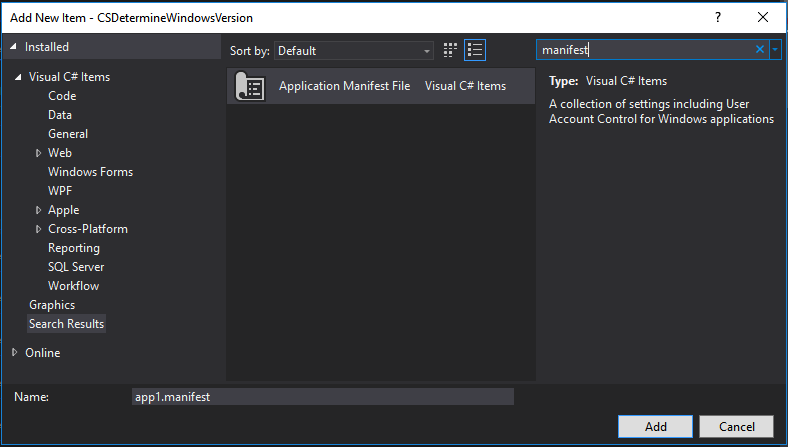
## Running the sample

* Open the sample solution using Visual Studio, then press **F5 Key** or select **Debug -> Start Debugging** from the menu.
* When the console application is running, you can see the result in the console window.



## Using the code

Add the manifest file to your application:



Config the manifest file like this:

<assembly manifestVersion="1.0" xmlns="urn:schemas-microsoft-com:asm.v1">

……

<compatibility xmlns="urn:schemas-microsoft-com:compatibility.v1">

<application>

<!-- A list of the Windows versions that this application has been tested on and is

is designed to work with. Uncomment the appropriate elements and Windows will

automatically selected the most compatible environment. -->

<!-- Windows Vista -->

<supportedOS Id="{e2011457-1546-43c5-a5fe-008deee3d3f0}" />

<!-- Windows 7 -->

<supportedOS Id="{35138b9a-5d96-4fbd-8e2d-a2440225f93a}" />

<!-- Windows 8 -->

<supportedOS Id="{4a2f28e3-53b9-4441-ba9c-d69d4a4a6e38}" />

<!-- Windows 8.1 -->

<supportedOS Id="{1f676c76-80e1-4239-95bb-83d0f6d0da78}" />

<!-- Windows 10 -->

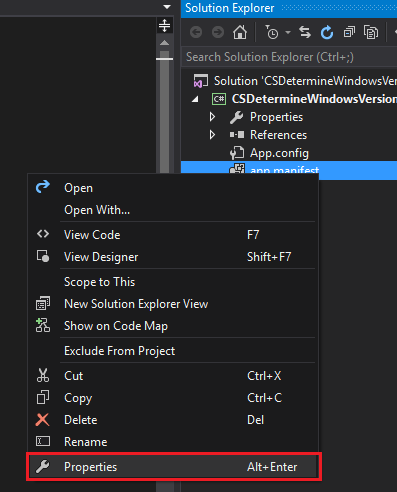
<supportedOS Id="{8e0f7a12-bfb3-4fe8-b9a5-48fd50a15a9a}" />

</application>

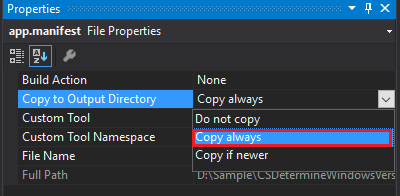
</compatibility>

</assembly>

Right click the manifest file and choose properties.

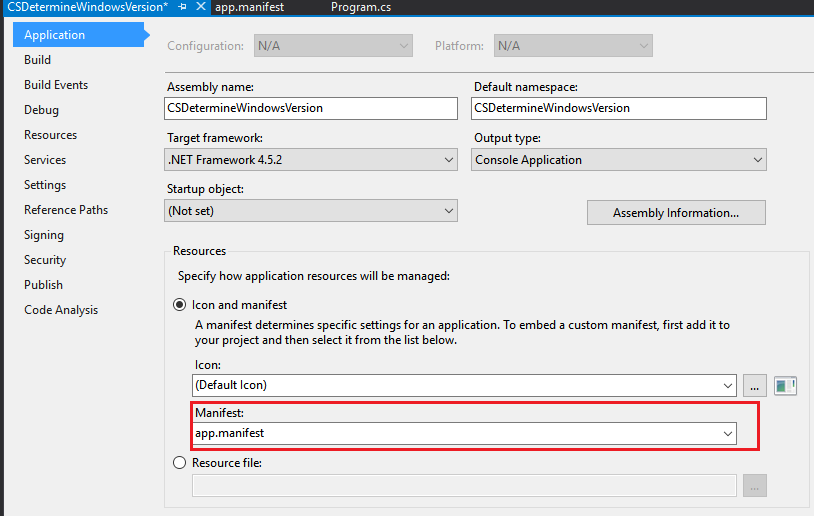


Set **Copy to Output Directory** as **Copy always**.



Right click your project and choose **Properties**.

In Application tab, ensure that have been met, **Resources -> Icon and manifest -> Manifest** set to you manifest file.



The OSVersionInfo Class Code:

public class OSVersionInfo

{

public string Name { get; set; }

public string FullName

{

get

{

return "Microsoft " + Name + " " + "[Version " + Major + "." + Minor + "." + Build + "]";

}

}

public int Minor { get; set; }

public int Major { get; set; }

public int Build { get; set; }

private OSVersionInfo() { }

/// <summary>

/// Init OSVersionInfo object by current windows environment

/// </summary>

/// <returns></returns>

public static OSVersionInfo GetOSVersionInfo()

{

System.OperatingSystem osVersionObj = System.Environment.OSVersion;

OSVersionInfo osVersionInfo = new OSVersionInfo()

{

Name = GetOSName(osVersionObj),

Major = osVersionObj.Version.Major,

Minor = osVersionObj.Version.Minor,

Build = osVersionObj.Version.Build

};

return osVersionInfo;

}

/// <summary>

/// Get current windows name

/// </summary>

/// <param name="osInfo"></param>

/// <returns></returns>

static string GetOSName(System.OperatingSystem osInfo)

{

string osName = "unknown";

switch (osInfo.Platform)

{

//for old windows kernel

case System.PlatformID.Win32Windows:

osName = ForWin32Windows(osInfo);

break;

//fow NT kernel

case System.PlatformID.Win32NT:

osName = ForWin32NT(osInfo);

break;

}

return osName;

}

/// <summary>

/// for old windows kernel

/// this function is the child function for method GetOSName

/// </summary>

/// <param name="osInfo"></param>

/// <returns></returns>

static string ForWin32Windows(System.OperatingSystem osInfo)

{

string osVersion = "Unknown";

//Code to determine specific version of Windows 95,

//Windows 98, Windows 98 Second Edition, or Windows Me.

switch (osInfo.Version.Minor)

{

case 0:

osVersion = "Windows 95";

break;

case 10:

switch (osInfo.Version.Revision.ToString())

{

case "2222A":

osVersion = "Windows 98 Second Edition";

break;

default:

osVersion = "Windows 98";

break;

}

break;

case 90:

osVersion = "Windows Me";

break;

}

return osVersion;

}

/// <summary>

/// fow NT kernel

/// this function is the child function for method GetOSName

/// </summary>

/// <param name="osInfo"></param>

/// <returns></returns>

static string ForWin32NT(System.OperatingSystem osInfo)

{

string osVersion = "Unknown";

//Code to determine specific version of Windows NT 3.51,

//Windows NT 4.0, Windows 2000, or Windows XP.

switch (osInfo.Version.Major)

{

case 3:

osVersion = "Windows NT 3.51";

break;

case 4:

osVersion = "Windows NT 4.0";

break;

case 5:

switch (osInfo.Version.Minor)

{

case 0:

osVersion = "Windows 2000";

break;

case 1:

osVersion = "Windows XP";

break;

case 2:

osVersion = "Windows 2003";

break;

}

break;

case 6:

switch (osInfo.Version.Minor)

{

case 0:

osVersion = "Windows Vista";

break;

case 1:

osVersion = "Windows 7";

break;

case 2:

osVersion = "Windows 8";

break;

case 3:

osVersion = "Windows 8.1";

break;

}

break;

case 10:

osVersion = "Windows 10";

break;

}

return osVersion;

}

}

Use the class:

///Notice that if you do not config the manifest file, you can't get the win10 version info

var osInfo = OSVersionInfo.GetOSVersionInfo();

Console.WriteLine(osInfo.FullName);

## More information

How to determine the Windows version by using Visual C#

<https://support.microsoft.com/en-us/kb/304283>

Targeting your application for Windows

<https://msdn.microsoft.com/en-us/library/windows/desktop/dn481241(v=vs.85).aspx>