# How do I get the computer's serial number? Consuming Windows Runtime classes in desktop apps, part 4: C#

devblogs.microsoft.com/oldnewthing/20180109-00

January 9, 2018

Raymond Chen

Continuing our series on getting the computer's serial number in desktop apps in various languages, we look at C#.

From Visual Studio, create a new C# Console Application that goes like this:

```
class Program
{
  static public void Main()
  {
    var serialNumber = Windows.System.Profile.SystemManufacturers.
                       SmbiosInformation.SerialNumber;
    System.Console.WriteLine($"Serial number = {serialNumber}");
  }
}
```

Before building, you'll have to prepare the project, and the preparation is particularly ugly.

- Close the solution in Visual Studio and open the `*.csproj` file in a text editor.
- Add

  ```
  <TargetPlatformVersion>8.0</TargetPlatformVersion>
  ```

  to the main `PropertyGroup`. This requirement is obscurely documented on MSDN. Scott Hanselman tipped me off.
- Reopen the project, right click the References node and select *Add Reference*.
- The magic XML you added to the `*.csproj` enables a new node in the dialog box called *Windows*. Expand it, click on *Core*, and then check `Windows. System` because we are using `Windows. System. BlahBlah`. In general, check each second-level namespace your program uses.

Adding a reference from *Core* will access the information from your development machine, so it assumes that your development machine is running the same or greater version of Windows than your target. If you are doing cross-targeting, then instead of referencing the `Windows. Blah`

namespaces under *Core*, go to the *Browse* option and browse to `C:\ Program Files (x86)\ Windows Kits\ 10\ References\ CONTRACT\ VERSION\ CONTRACT.winmd`.

In our case, `Windows. System. Profile. SystemManufacturers. SmbiosInformation` is in the `Windows. System. Profile. SystemManufacturers. SystemManufacturersContract` contract. I got this information from the documentation for the `SmbiosInformation` class: Look under **API contract**.

That documentation also says that the `SmbiosInformation` class was introduced in v1, so the minimum version I need is 1.0.0.0. The full path is therefore

```
C:\Program Files (x86)\
  Windows Kits\
    10\
      References\
        Windows.System.Profile.SystemManufacturers.SystemManufacturersContract\
          1.0.0.0\
            Windows.System.Profile.SystemManufacturers.SystemManufacturersContract.winmd
```

Repeat for each contract your program requires. Most classes are in the `Windows. Foundation. UniversalApiContract` contract.

The last bit is another obscure piece of information on MSDN: Adding a reference to `System.Runtime`. If you use a Windows Runtime class that projects as an `IDictionary` or some other fancy type, then you will get the error message "The type 'IDictionary`2' is defined in an assembly that is not referenced. You must add a reference to assembly 'System.Runtime, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a'." But when you go to the *Add Reference* dialog, you won't see it! It's telling me to add a reference to an assembly that doesn't exist!

That's because it's hidden away somewhere that Visual Studio doesn't show you. Go to the *Browse* tab, click the Browse button, and then go to `%ProgramFiles(x86)%\ Reference Assemblies\ Microsoft\ Framework. NETFramework\ v4.5\ Facades\ System.Runtime.dll`, substituting the version of the .NET Framework that applies to your project.

Okay, now that you got all the references set up, you can build and run the program.

It takes some work to set up, but personally I find C# to be the most convenient way of consuming Windows Runtime classes.

Next up is PowerShell. Just warning you ahead of time: You're going to be underwhelmed.

Raymond Chen

**Follow**