How do I get the computer's serial number? Consuming Windows Runtime classes in desktop apps, part 2: C++/CX

devblogs.microsoft.com/oldnewthing/20180105-00

January 5, 2018

Raymond Chen

Continuing our series on getting the computer's serial number in desktop apps in various languages, next up is C++/CX.

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

```
#include <windows.h>
#include <stdio.h> // Horrors! Mixing C and C++!

[Platform::STAThread]
int __cdecl wmain(int, wchar_t**)
{
  CCoInitialize init;

  auto serialNumber = Windows::System::Profile::SystemManufacturers::
                      SmbiosInformation::SerialNumber;
  wprintf(L"Serial number = %ls\n", serialNumber->Data());

  return 0;
}
```

Before building, right-click the Project in Visual Studio and select *Properties*, and then make these changes:

- Configuration Properties, C/C++, General, Additional `#using` Directories: Add these two directories, adjusting as appropriate for where you installed Visual Studio and the Windows SDK.
  - `C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\vcpackages` (so the compiler can find `platform.winmd`)
  - `C:\Program Files (x86)\Windows Kits\10\UnionMetadata\10.0.16299.0` (so the compiler can find `windows.winmd`)[1]
- Configuration Properties, C/C++, General, Consume Windows Runtime Extension: Set to **Yes (/ZW)**.

- Configuration Properties, C/C++, Code Generation, Enable Minimal Rebuild: Set to **No (/Gm-)**.
- Configuration Properties, Linker, Inputs, Additional Dependencies: add `windowsapp.lib`.

Okay, now you can build and run the program.

Consuming Windows Runtime objects in C++/CX is more convenient than accessing them raw, but it is a nonstandard Microsoft extension.

You don't have to build your entire application in C++/CX. You can write part of it in plain C++, and part of it in C++/CX, and the link the two pieces together. The Casting page on MSDN explains how to convert between a hat-pointer and a regular pointer.

Okay, so setting up the project was kind of ugly, but that's okay, because things will get better before they get better. Up next is C++/WinRT.

[1] There are two copies of `windows.winmd`, a good one in the directory I gave above, and a bad one in the directory `UnionMetadata\ Facade`. If you use the bad one, you get an internal compiler error. Larry Osterman tried to explain to me what the bad copy in Facade was for, but all I heard was the wah-wah of Charlie Brown's teacher.

Raymond Chen

Follow