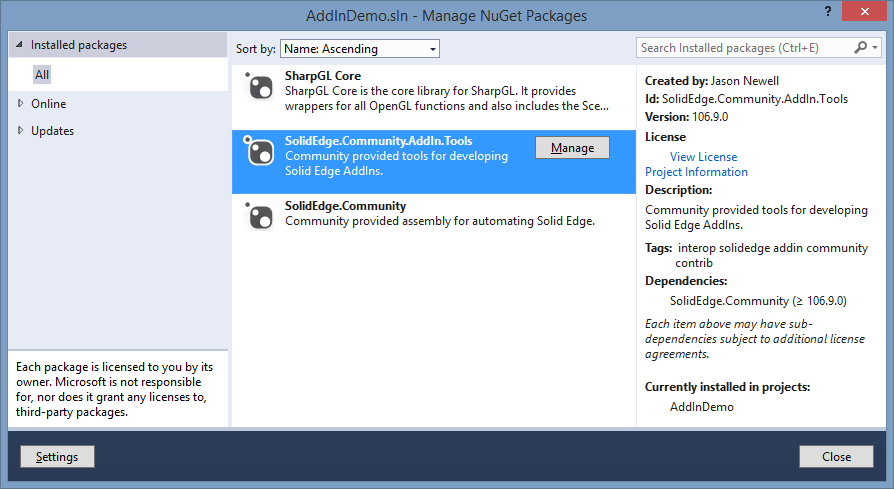
Solid Edge AddIn Demo

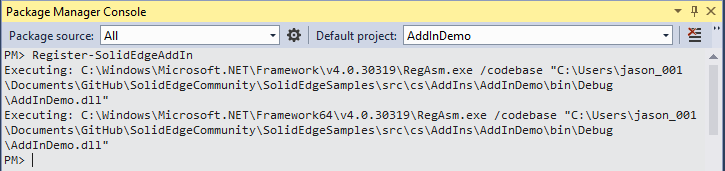
# NuGet

This project makes use of open source projects via [NuGet](https://www.nuget.org/). The NuGet packages used are shown below. Of particular interest is the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. It has a dependency on [SolidEdge.Community](https://www.nuget.org/packages/SolidEdge.Community). It also installs package manager console commands and post build project events that enable you to embed images as native resources into the assembly.

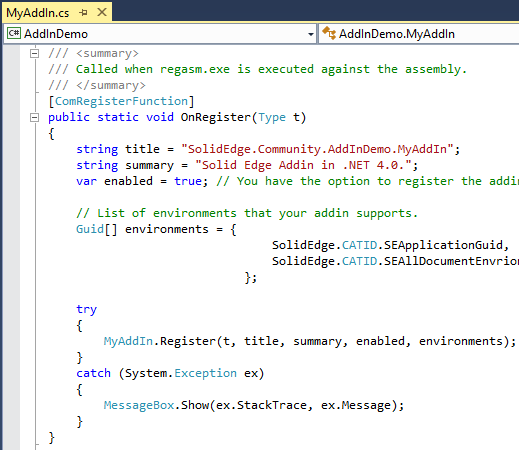


# Registration

Registering your addin has never been easier due to the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. In Visual Studio 2012\2013, navigate to Tools -> NuGet Package Manager -> Package Manager Console. Type ***Register-SolidEdgeAddIn*** and hit enter. This command will execute the x86 & x64 version of regasm.exe against your assembly.

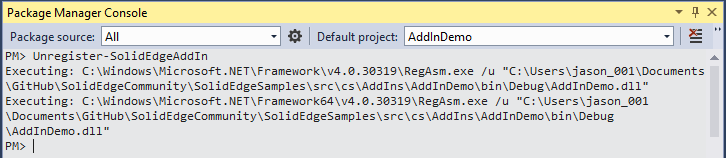


When regasm.exe is executed against the assembly, the **OnRegister** method will get called. This is where you can control the registration process of your addin.

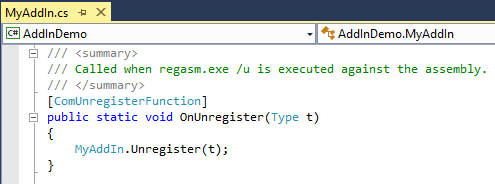


# Unregistration

Unregistering the addin is just as easy. In Visual Studio 2012\2013, navigate to Tools -> NuGet Package Manager -> Package Manager Console. Type **Unr*egister-SolidEdgeAddIn*** and hit enter. This command will execute the x86 & x64 version of regasm.exe /u against your assembly.

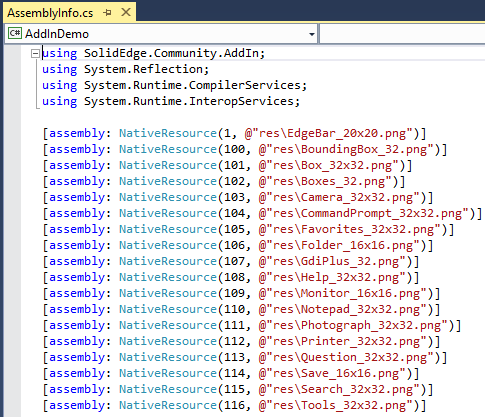


When regasm.exe /u is executed against the assembly, the **OnUnregister** method will get called. This is where you can control the unregistration process of your addin.

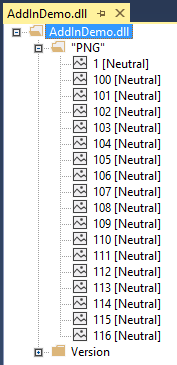


# Native Win32 Resources

Embedding native Win32 resources for your EdgeBar and Ribbon is made possible by the [SolidEdge.Community.AddIn.Tools](https://www.nuget.org/packages/SolidEdge.Community.AddIn.Tools) package. Add your BMP\PNG resources to a folder under the project directory and modify the AssemblyInfo.(cs|vb) with the NativeResource attribute as shown below. After you build the solution, the resources will automatically get embedded into your .dll.



You can open your .dll in Visual Studio, Resource Hacker, etc. to verify that the resources got embedded during the post build event. If you do open your .dll in this manner, be sure to close it before attempting another build or you will get a build error due to the file being in use.

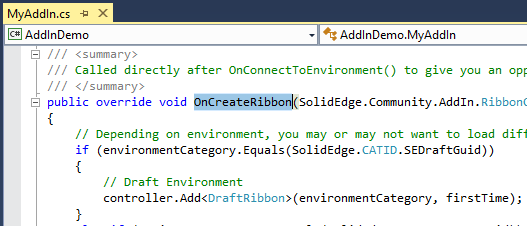


# Ribbons

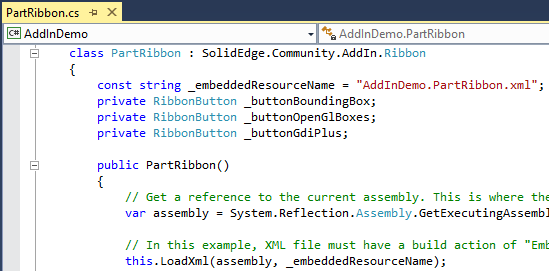
You can now completely define your Ribbon in XML as shown below. The imageId attribute for each control points to a NativeResource that you previously defined in AssemblyInfo.(cs|vb).



The **OnCreateRibbon** method of your addin will get called when it’s time to create the Ribbon.

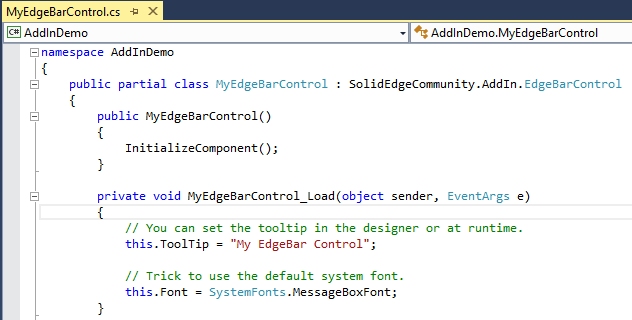


In the following example, an instance of our PartRibbon class get created. In the constructor, we load the Ribbon XML to configure the ribbon.

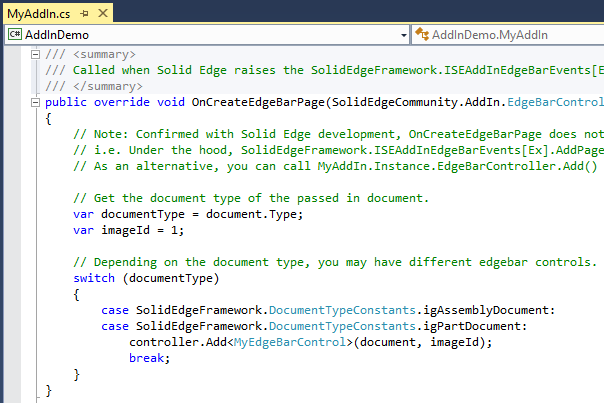


# EdgeBar

The SolidEdgeCommunity.AddIn.EdgeBarControl class provides the base functionality to create an EdgeBar control. Simply add a new “User Control” to your project and change the base class from UserControl to SolidEdgeCommunity.AddIn.EdgeBarControl.



When it’s time to add your control to the EdgeBar, the addin framework will call **OnCreateEdgeBarPage**.

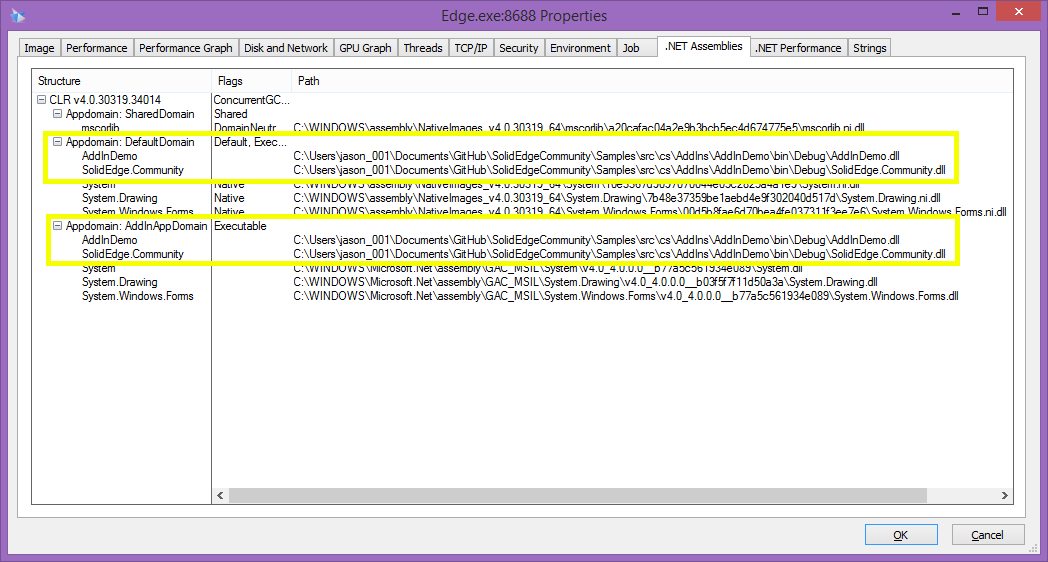


# AppDomains

One of the challenges of developing .NET Solid Edge AddIns in the past was the fact that every addin got loaded into the “DefaultDomain (AppDomain)”. This meant that every .NET addin shared the same runtime callable wrappers (RCW). If there are multiple .NET addins installed on the machine, the risk of a single addin causing problems for the other addins greatly increases.

The SolidEdgeCommunity.AddIn.SolidEdgeAddIn base class (mostly) fixes this issue by creating an isolated AppDomain for your addin to execute. Due to the current architecture of Solid Edge addins, we cannot avoid the fact that our addin will get loaded into the “DefaultDomain” but if you inherit from SolidEdgeCommunity.AddIn.SolidEdgeAddIn, you minimize the possibility of your addin being affected by other .NET addins.

The following screenshot is from Process Explorer. The current url is <http://technet.microsoft.com/sysinternals/bb896653.aspx> but may change.



# Deployment (manual)

There are various method to deploying a Solid Edge AddIn. In its simplest form, you simply copy the required files to the target machine and execute RegAsm.exe on the addin.dll. RegAsm.exe does nothing more than write required registry entries that allow Solid Edge to see and load the addin.dll.

* Copy AddInDemo.dll & SolidEdge.Community.dll to target machine (same folder).
* C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe /codebase [path\_to\_addin.dll]
  + On an x64 machine, registers for Solid Edge x64.
  + On an x86 machine, registers for Solid Edge x86.
* C:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe /codebase [path\_to\_addin.dll]
  + On an X64 machine, registers for Solid Edge x86.

The following screenshot demonstrates how to view the registry entries. You can search by your addin guid. In this case, BF1C1BB8-75EE-444A-8DCE-0F1521D0764B.

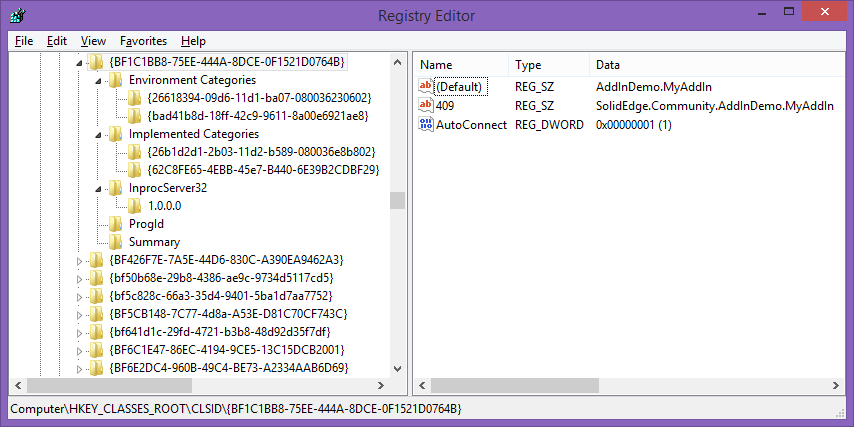


Figure - Windows x64 - Solid Edge x64

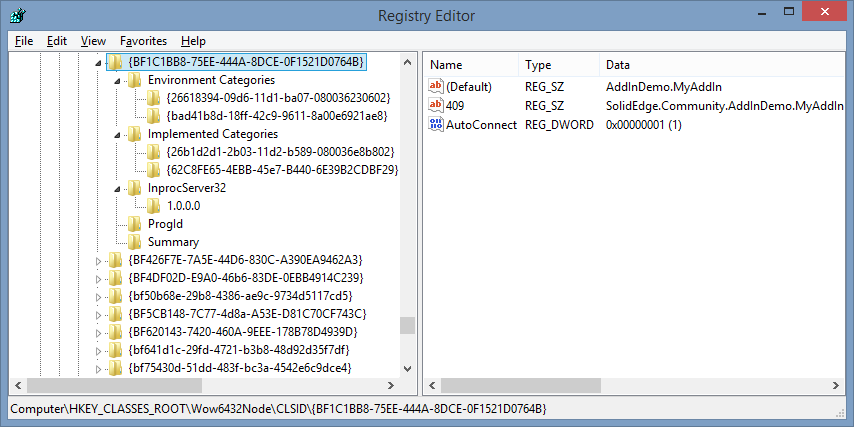


Figure - Windows x64 - Solid Edge x86

# Deployment (installer)

There are various installer products available so it’s not realistic to cover them all. The general concept is basically the same as the manual approach. Most modern installers offer you the ability to record and embed the registry entries so that calling RegAsm.exe is not necessary. In most cases, that is a desirable feature and the recommended approach if available.