How to create a custom install program

Summary

To deploy stand-alone applications (.exe) or in-process components (.dll), you can create a custom installation program using Visual Studio or third-party installation software such as Wise or InstallShield. By using Visual Studio, it is possible to create a custom install program that deploys and registers your components—the key being the System.Runtime.InteropServices.RegistrationServices class. This class has the RegisterAssembly and UnregisterAssembly members, which register and unregister managed classes with component object models (COM). These are the same functions the RegAsm utility uses. Using these functions inside a custom installer class along with a setup program is the complete solution.

Development licensing	**Deployment licensing**
Engine Developer Kit	Engine Runtime
ArcView	ArcView
ArcEditor	ArcEditor
ArcInfo	ArcInfo

Deploying a .NET component customization

The steps in this article show how to deploy a .NET component customization; for example, a class library developed in .NET that provides custom commands and tools.

The technique described in this article assumes that ArcGIS is installed on the target machine.

1. In Visual Studio, open a .NET solution that already contains a class library project with at least one class exposed to COM; for example, a custom command class. You may want to create your command using Visual Studio IDE Integration features of the .NET SDK.

For more information on using the Visual Studio IDE integration to create you own custom ArcGIS based COM components see About the ArcGIS item templates.

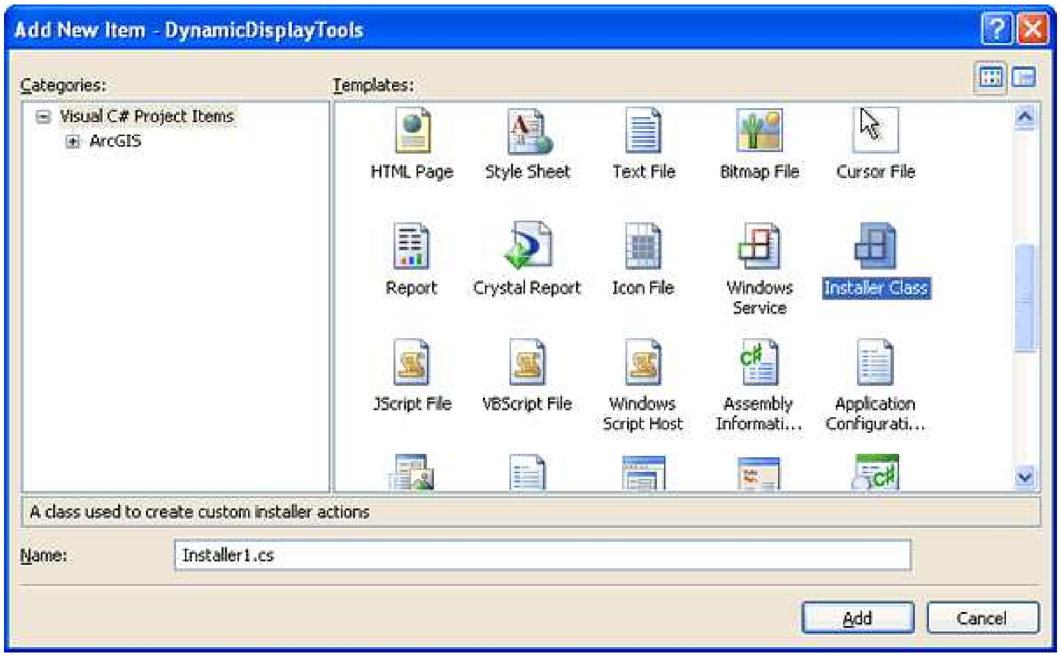
2. In the Solution Explorer, right click on the project name and choose 'Add' > 'New Item' and then choose 'Installer Class' from the *New Item* dialog. See the following screen shot:

Search

Documentation Library

GO

Search Help



A class used to create custom installer actions

Name: Installer1.cs

You will need to have a name for your class other than "Installer" as that is the name of the class in System.Configuration.Install that your install class will inherit from.

3. The next step is to override the install and uninstall functions implemented in the installer base class. You use the RegistrationServices class RegisterAssembly and UnregisterAssembly methods to register the components. Make sure you use the SetCodeBase flag to indicate that the code base key for the assembly is set in the registry. The RegistrationServices class is found in the System.Runtime.InteropServices namespace. To accomplish this, first add a new imports or using statement at the top of the class. See the following:

[C#]

```
using System.Runtime.InteropServices;
```

[VB.NET]

```
Imports System.Runtime.InteropServices
```

Then, add the following code inside of the installer class:
[C#]

```
public override void Install(System.Collections.IDictionary stateSaver)
{
  base.Install(stateSaver);
  RegistrationServices regSrv = new RegistrationServices();
  regSrv.RegisterAssembly(base.GetType().Assembly,
    AssemblyRegistrationFlags.SetCodeBase);
}

public override void Uninstall(System.Collections.IDictionary savedState)
{
  base.Uninstall(savedState);
  RegistrationServices regSrv = new RegistrationServices();
  regSrv.UnregisterAssembly(base.GetType().Assembly);
}
```

[VB.NET]

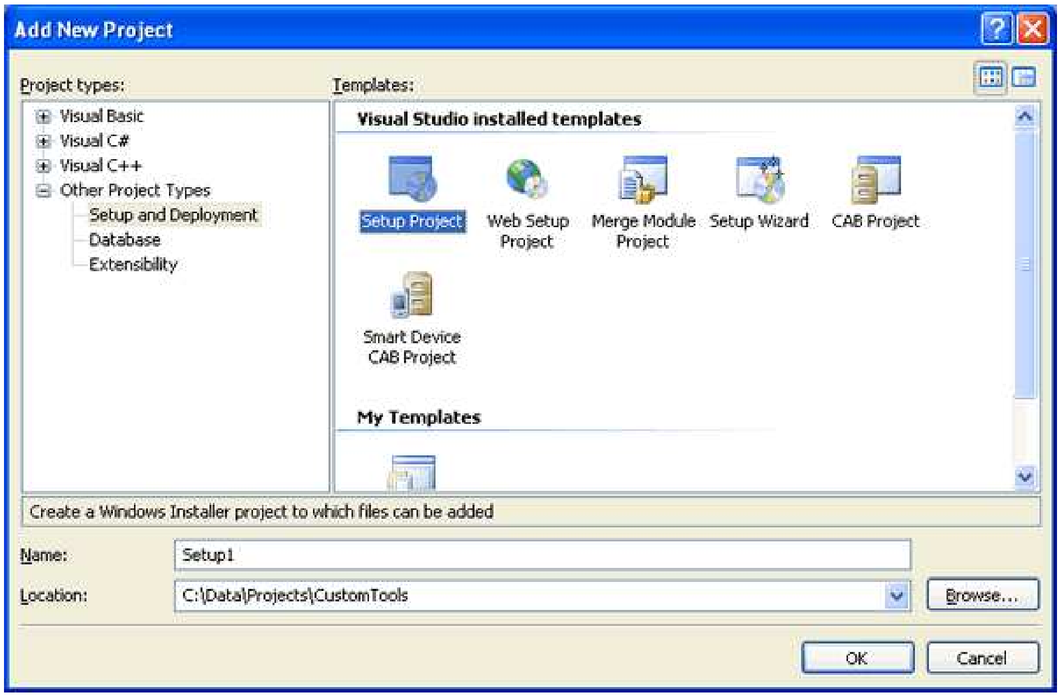
```
Public Overrides Sub Install(ByVal stateSaver As System.Collections.IDictionary)
MyBase.Install(stateSaver)
Dim regsrv As New RegistrationServices
regsrv.RegisterAssembly(MyBase.GetType().Assembly, AssemblyRegistrationFlags.SetCodeBase)
End Sub
```

```
Public Overrides Sub Uninstall(ByVal savedState As System.Collections.IDictionary)
MyBase.Uninstall(savedState)
Dim regsrv As New RegistrationServices
regsrv.UnregisterAssembly(MyBase.GetType().Assembly)
End Sub
```

4. At this point you must re-build your project to ensure that the latest version of all the project files get included in the setup program. Choose Build > Rebuild Solution from the Visual Studio menus.

If you want your application to be able to work across different versions of assemblies, you may need to specify that in the properties of any referenced assemblies. You can do this by clicking on a specific assembly in the References section of the Solution Explorer and setting the 'Specific Version' to False. Do this before you re-build your project.

5. Add a setup program to your solution by choosing 'File' > 'New Project' from the menus. On the Add New Project dialog box, select Setup and Deployment in the Project types area, then click Setup Project in the Templates area. Name the new project and click OK to add the new setup project to your solution. See the following screen shot:

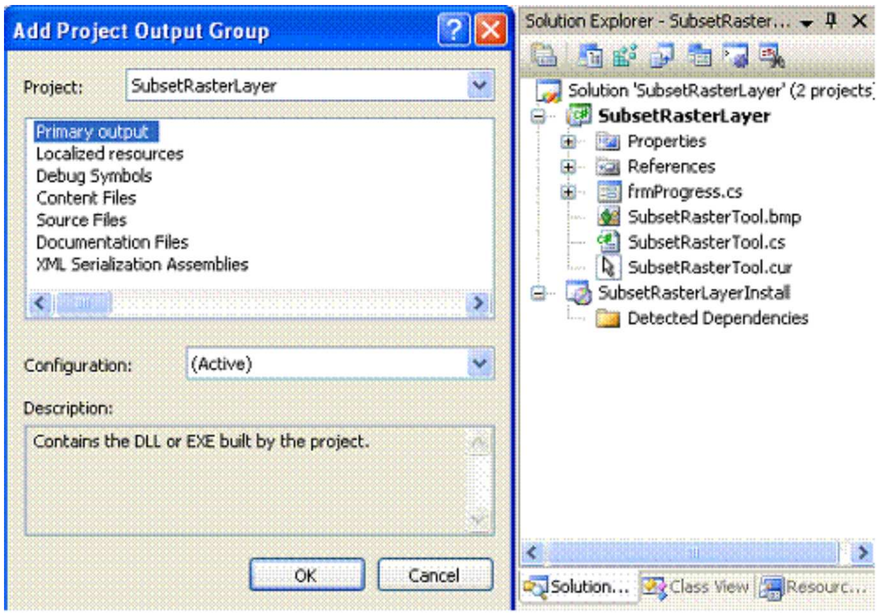


Create a Windows Installer project to which files can be added

Name: Setup1

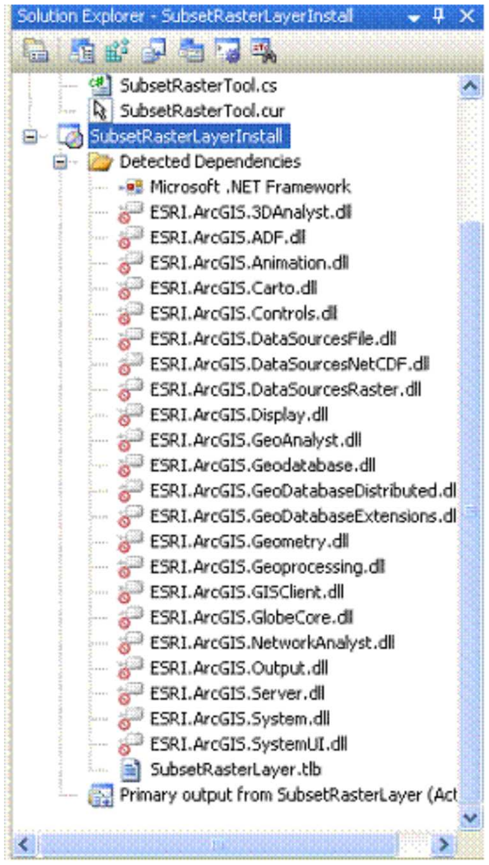
Location: C:\Data\Projects\CustomTools

6. In the Solution Explorer, right-click the new Setup project, click Add, then select Project Output. Choose the project you want to deploy. Select Primary output and click OK. See the following screen shot:



Contains the DLL or EXE built by the project.

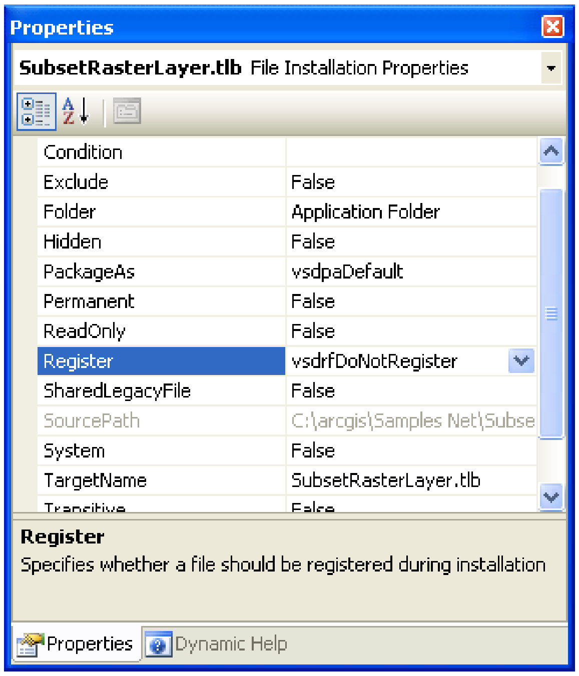
7. From the list of detected dependencies that is regenerated, exclude all references to ESRI primary interop assemblies; for example, ESRI.ArcGIS.System. Also exclude the Stdole.dll reference. The only items typically left in the list are the .tlb file, Microsoft .NET Framework file, and the Primary output from the <AssemblyName><Version> file, which represents the .dll or .exe you are compiling. See the following screen shot:


Solution Explorer - SubsetRasterLayerInstall
SubsetRasterTool.cs
SubsetRasterTool.cur
SubsetRasterLayerInstall
Detected Dependencies
Microsoft .NET Framework
ESRI.ArcGIS.3DAnalyst.dll
ESRI.ArcGIS.ADF.dll
ESRI.ArcGIS.Animation.dll
ESRI.ArcGIS.Carto.dll
ESRI.ArcGIS.Controls.dll
ESRI.ArcGIS.DataSourcesFile.dll
ESRI.ArcGIS.DataSourcesNetCDF.dll
ESRI.ArcGIS.DataSourcesRaster.dll
ESRI.ArcGIS.Display.dll
ESRI.ArcGIS.GeoAnalyst.dll
ESRI.ArcGIS.Geodatabase.dll
ESRI.ArcGIS.GeoDatabaseDistributed.dl
ESRI.ArcGIS.GeoDatabaseExtensions.dl
ESRI.ArcGIS.Geometry.dll
ESRI.ArcGIS.Geoprocessing.dll
ESRI.ArcGIS.GISClient.dll
ESRI.ArcGIS.GlobeCore.dll
ESRI.ArcGIS.NetworkAnalyst.dll
ESRI.ArcGIS.Output.dll
ESRI.ArcGIS.Server.dll
ESRI.ArcGIS.System.dll
ESRI.ArcGIS.SystemUI.dll
SubsetRasterLayer.tlb
Primary output from SubsetRasterLayer (Act

You should never include ArcGIS assemblies in your deployment package as these are installed by the ArcGIS installation program.

8. Additionally, you need to change the value of the Register property for the .tlb file. By default, the .tlb file is set to be registered with COM at install time. However, this causes an error during installation. To accomplish this, right-click the .tlb file in the project's detected dependencies tree node. In the Properties pane, set the value of the Register property to vsdrfDoNotRegister before

compiling the installer. See the following screen shot:



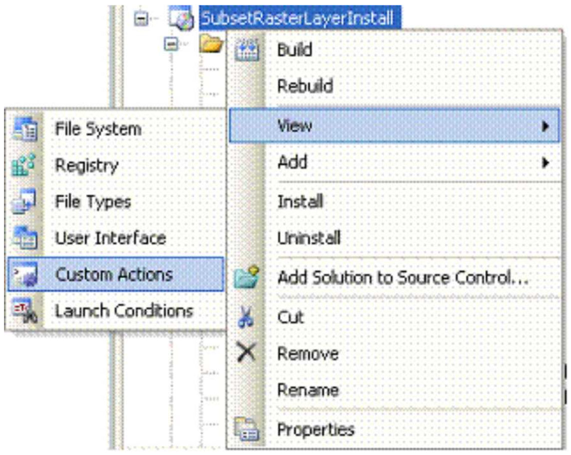
Properties

SubsetRasterLayer.tlb File Installation Properties

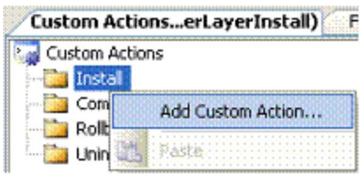
Condition	
Exclude	False
Folder	Application Folder
Hidden	False
PackageAs	vsdpaDefault
Permanent	False
ReadOnly	False
Register	vsdrfDoNotRegister
SharedLegacyFile	False
SourcePath	C:\arcgis\Samples Net\Subse
System	False
TargetName	SubsetRasterLayer.tlb
Transitive	False

Register
Specifies whether a file should be registered during installation

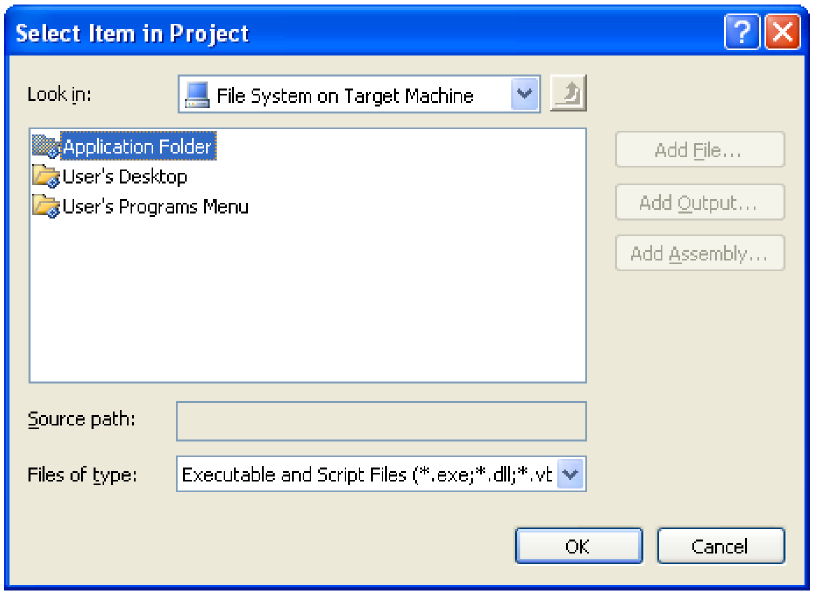
9. The final steps involve associating the custom installation steps configured in the new installer class with the setup project. To do this, right-click the setup project in the Solution Explorer, choose View, then choose Custom Actions. See the following screen shot:



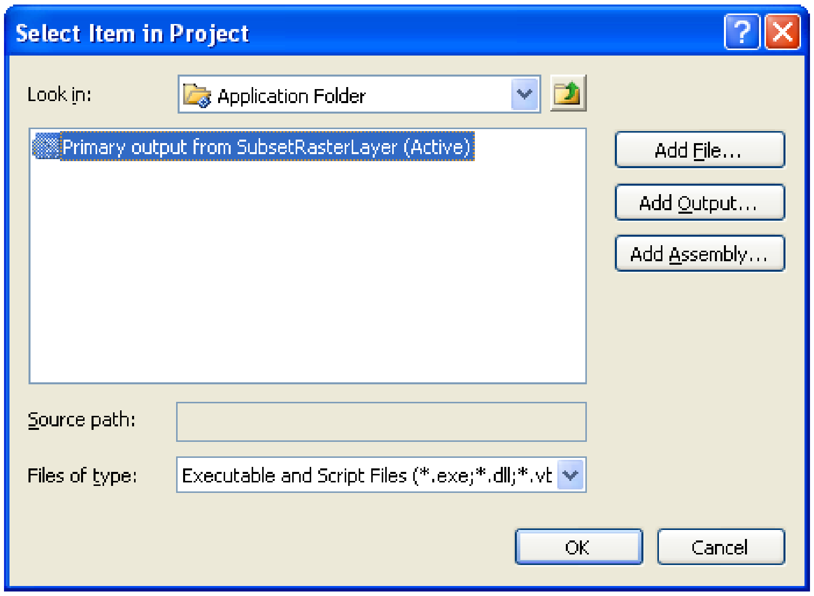
10. In the resulting custom actions for the installation project, right-click the Install folder and choose Add Custom Action. See the following screen shot:



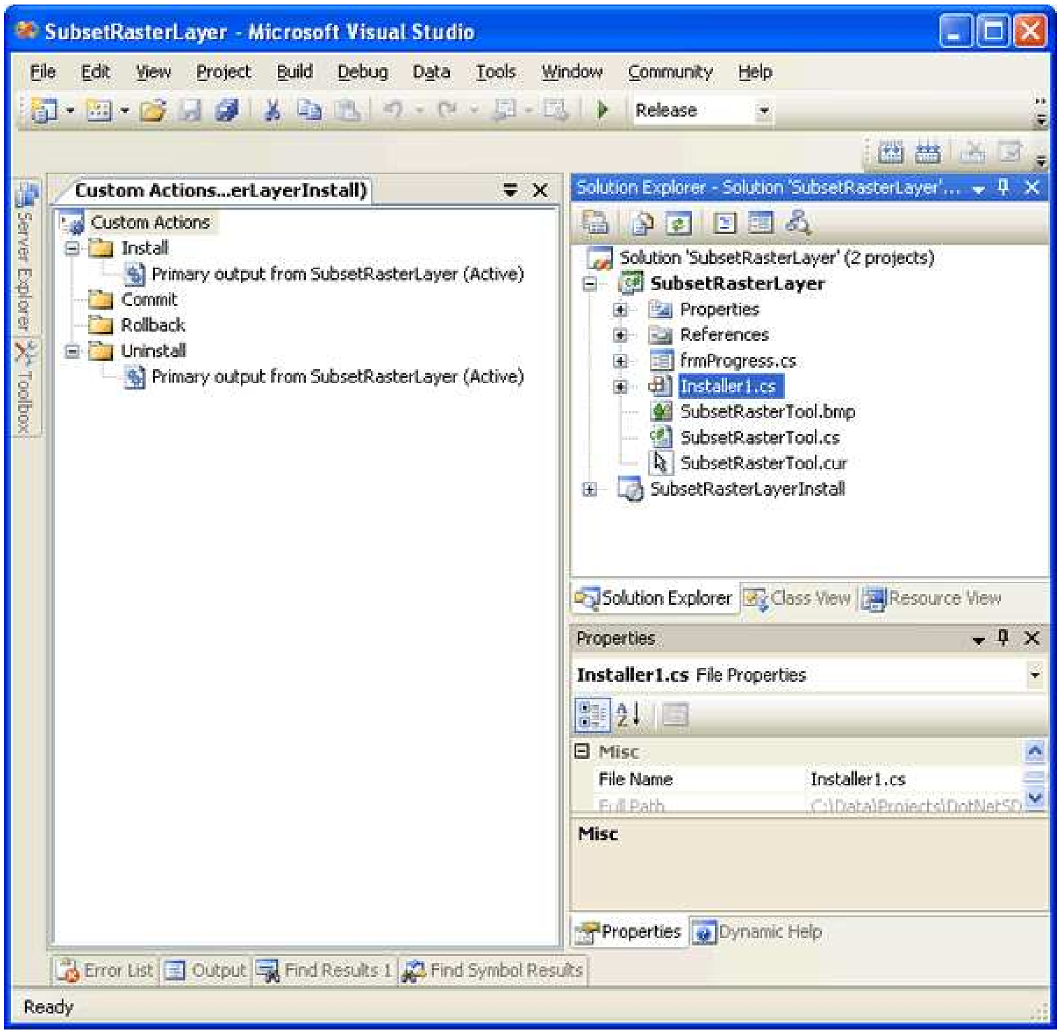
11. On the Select Item in Project dialog box, double-click Application Folder. See the following screen shot:



12. From the Application Folder, double-click the Primary output from the <AssemblyName><Version> file to associate the custom install function created earlier with the setup's custom install action. See the following screen shot:



13. Repeat this process for the setup's uninstall. See the following screen shot:



14. Rebuild the entire solution to generate the setup's executable.

> A setup project is not built by default. In the Solution Explorer, you can right-click the setup project and click Build to build the setup project. You can also right-click the solution and choose Configuration Manager, then on the Configuration Manager dialog box, select the Build option for the setup project before rebuilding the solution.

Running the setup executable on a target machine installs the components and registers them with COM. The COMRegisterFunction routine then registers the components in the appropriate component categories.

> Please note that you must uninstall the custom component before you attempt to uninstall ArcGIS. The reason for that is that custom components need to loaded at install/uninstall time. When loading the .NET Framework loads the custom component it must also load all referenced assemblies (eg "ESRI.ArcGIS.ArcMap.dll" and "ESRI.ArcGIS.System.dll"). If these referenced assemblies are missing at install/uninstall time then the install/uninstall will fail.

See Also:

How to register .NET components with COM
How to register classes in COM component categories
About the ArcGIS item templates

User comments | Add your comment

Would you please post the complete Installer code file (C#) posted by: Will_Branch on Sep 24, 2007 14:40

http://forums.esri.com/Thread.asp?c=159&f=2267&t=234774#714224

Re: Would you please post the complete Installer code file posted by: kylie on Oct 01, 2007 10:10

Working with the user, we found that the issues he was facing were due to a naming conflict. We have updated this doc with a warning about naming when creating your installer class. -- Kylie, ESRI SDK Team

---

Well done. posted by: cpalmer on Oct 23, 2007 14:37

This is the way help topics are supposed to look. Excellent step-by-step instructions that were easy to follow. Now all I have to do is get the setup.exe to work.

Would you please post the complete Installer code file (VB .NET) posted by: xzhang_usa on May 08, 2008 06:57

Would you please post the complete Installer code file (VB .NET)

AI posted by: Maksimus1200 on Feb 08, 2009 08:07

Also you can make it easily with Actual Installer

Appreciated posted by: santoshf2 on Aug 10, 2009 03:46

But i want to also add uninstall in the programs menu now how is that Santosh V santhosh@pixelsoftek.com

---

extra code needed posted by: artcoding on Sep 24, 2009 00:40

Please post c# code to call MsiSetTargetPath using dllimport to change the default installation destination path at runtime, as the function header does not tell too much.

This works great! posted by: brianl57 on Jan 12, 2010 11:37

Following these instructions I was able to to install an auto updater written in C# with absolutely no issues involving the installation. Thank you for the clear, concise, step-by-step instructions.

Little problem in installation... posted by: yashika.sareen on Oct 07, 2010 02:29

when i perform the above steps.....after implementing step 5 and 6 that is adding the setup and then selecting the project output...i noticed that the dependencies folder does not contain the tlb file neither did it contain Stdole.dll.Please advice how can i proceed further...its realy urgent

---