ADD-Ins Setup Creation for MS Office 2007 and Visual Basic 2010

**Deployment Methods:**

* **Clickonce**
* **Windows installer**

**Deploying Visual Studio Tools for the Office system 3.0 solutions**

Both Clickonce and Windows Installer packages need to perform the same rudimentary tasks when installing a VSTO 3.0 Solution.

1. Install prerequisite components on the user computer.
2. Deploy the solution specific components.
3. For add-ins, create registry entries.
4. Trust the solution to allow it to execute.

**Required Prerequisite Components on the Target Computer**   
To run Visual Studio 2010 Tools for Office solutions, the following software must be available on the user computer.

* The 2007 Microsoft Office system or Microsoft Office 2010.
* The Microsoft .NET Framework  
  Visual Studio Tools for the Office system 2010 can work with the Microsoft .NET Framework 3.5 or  
  Microsoft .NET Framework 4.
* The Microsoft Visual Studio 2010 Tools for Office Runtime.  
  Visual Studio 2010 Tools for Office Runtime provides a runtime environment that manages add-ins  
  and document-level solutions.
* The primary interop assemblies for the 2007 Microsoft Office system or Microsoft Office 2010.
* Any utilities assemblies referenced by projects that target the .NET Framework 4.

**Solution Specific Components**   
The installer package must install the following components to the user computer.

* The Microsoft Office document, if you create a document-level solution.
* The customization assembly and any assemblies it requires.
* Additional components such as configuration files.
* The application manifest (.manifest).
* The deployment manifest (.vsto).

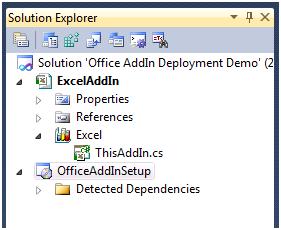
**Registry Entries for Add-ins**   
Add-ins require a set of registry entries for the Microsoft Office application to locate the add-in. You should create the registry entries as part of the deployment process. For more information about registry entries pertaining to a Visual Studio Tools for Office add-in, see [Registry Entries for Application-Level Add-Ins](http://msdn.microsoft.com/en-us/library/bb386106.aspx). When developing an Outlook add-in that displays custom form regions you should create additional registry keys as part of the installation procedure to allow the form regions to be identified. For more information about registry entries pertaining to Outlook form regions, see [Specifying Form Regions in the Windows Registry](http://msdn.microsoft.com/en-us/library/bb206787%28office.12%29.aspx).

**Trusting the Visual Studio Tools for Office Solution**   
Before you allow the customization to execute, a solution must be trusted. This trust relationship is based on the signed application and deployment manifests. You can either sign the manifests with a certificate that identifies a known and trusted publisher, or create a trust-relationship at installation time with an inclusion list entry or installing to a trusted location. For more information about how to obtain a certificate for signing, see [ClickOnce Deployment and Authenticode](http://msdn.microsoft.com/en-us/library/ms172240.aspx). For more information about using the user inclusion list, see [Trusting Office Solutions by Using Inclusion Lists](http://msdn.microsoft.com/en-us/library/bb608607.aspx).

CASE Study:

To create the AddInSetup project

1. Open the Office AddIn Project you want to deploy using Windows Installer.
2. With the Office Project Open, on the **File** menu, expand **Add** and click **New Project** to add a new project.
3. In the **Add New Project** dialog box, in **the Project types** pane, expand **Other Project Types** and then  
   expand **Setup and Deployment** and then select **Visual Studio Installer**.
4. In the **Templates** pane, select **Setup Project** from the **Visual Studio installed** templates group.
5. In the **Name** box, type **OfficeAddInSetup**.
6. Click **Open** to create the new setup project. Visual Studio opens the File Explorer for the new setup project. This explorer allows you to add files to the setup project from locations such as the projects inside your solution or loose files from your file-system.

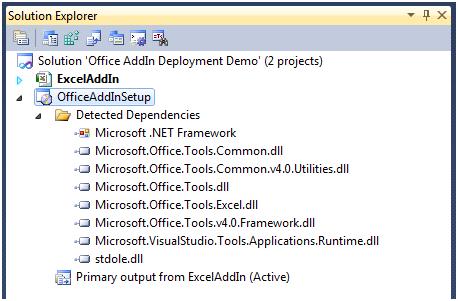


**Figure 1: Solution Explorer for the setup project**

The setup project needs to deploy the ExcelAddIn DLL. You can configure the setup project for this task by adding the ExcelAddIn project output to the setup project.

To add the ExcelAddIn project output

1. In the **Solution Explorer**, right-click **OfficeAddInSetup**, click **Add** and then **Project Output**.
2. In the **Add Project Output Group** dialog box, confirm that the **ExcelAddIn** project is selected, and the  
   **Primary Output** option is selected.
3. Click **OK** to add the project output to the setup project.

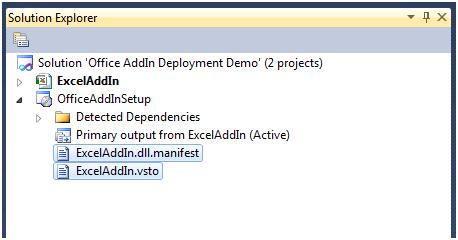


**Figure 2: Dependencies for the setup project**

The setup project needs to deploy the deployment manifest and application manifest. You can add these two files to the setup project as stand-alone files from the output folder of the ExcelAddIn project.

To add the deployment and application manifests

1. In the **Solution Explorer**, right-click **ExcelAddInSetup**, click **Add**, and click **File**.
2. In the **Add Files** dialog box, navigate to the **ExcelAddIn** output directory. Usually the output directory is the **bin\release** subfolder of the project root directory, depending on the selected build configuration.
3. Select the **ExcelAddIn.vsto** and **ExcelAddIn.dll.manifest** files and click **Open** to add these two files to the setup project.



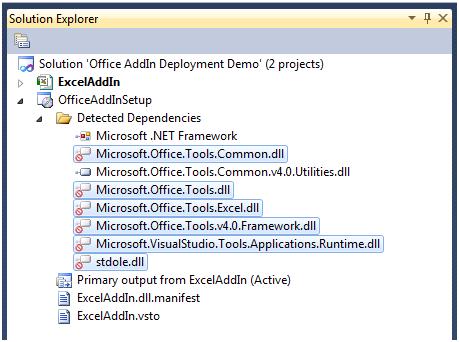
**Figure 3: Application and deployment manifests for the add-in**

The downside of adding loose files to a setup project is that you reference the default output folder of the Release build configuration explicitly. When you change the build configuration back to Debug, the setup project still references the files from this folder. Because you should only distribute release builds, this is acceptable. Alternatively, you can configure the Debug and Release build configuration to output their files to the same directory.

Referencing the ExcelAddIn includes all the components that ExcelAddIn requires. These components must be excluded and deployed using prerequisite packages to allow them to be registered correctly. Also, the Software License Terms must be displayed and accepted before the installation begins.

To exclude the ExcelAddIn project dependencies

1. In the **Solution Explorer**, in the **OfficeAddInSetup** node, select all dependency items beneath the **Detected Dependencies** item except for **Microsoft .NET Framework** or any assembly that ends with **\*.Utilities.dll**. The Utilities assembly is only present when your Office solution targets .NET 4 and the assembly is meant to be deployed along with your application.
2. Right-click the group and select **Properties**.
3. In the **Properties** window, change the **Exclude** property to **True** to exclude the dependent assemblies from the setup project. Make sure to not exclude any Utilities assemblies.



**Figure 4: Excluding dependencies**

You can configure your Windows Installer package to install the prerequisite components by adding a Setup program, also known as a bootstrapper. The Setup program can include and install the redistributables provided by the component vendors, a process called bootstrapping. For the ExcelAddIn, the following prerequisites must be installed before the add-in can run correctly.

* The Microsoft .NET Framework version that the Office Solution targets - .NET Framework 3.5 SP1, .NET  
  Framework 4 Client Profile, or .NET Framework 4.
* Microsoft Visual Studio 2010 Tools for Office Runtime.
* Windows Installer 3.1.
* If your solution targets .NET Framework 3.5, you will also have to redistributable Primary Interop  
  Assemblies for either the 2007 Microsoft Office system or Microsoft Office 2010.

To configure dependent components as prerequisites

1. In the **Solution Explorer**, right-click the **OfficeAddInSetup** project and select **Properties**.
2. The **OfficeAddInSetup Property Pages** dialog box appears.
3. Click **Prerequisites**.
4. In the **Prerequisites** dialog box, perform the following tasks.

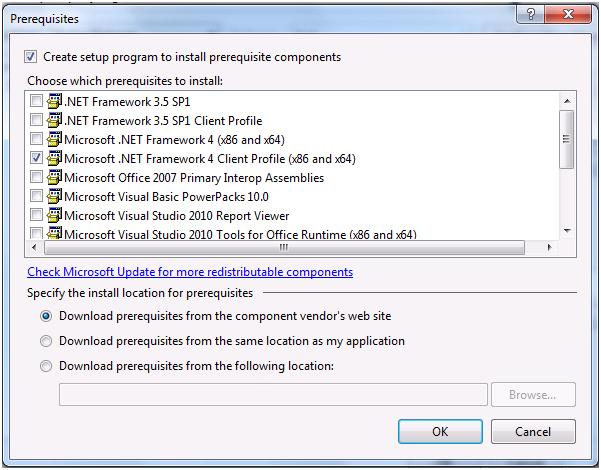
a.      Select **Create setup program to install prerequisite components**. This creates the setup.exe bootstrapper together with the MSI file.

b.      In the **Choose prerequisites to install** list, select the appropriate packages depending on what your solution targets.

c.      For **Specify the install location for prerequisites**, select **Download prerequisites from the component vendor’s web site.**

d.      Click **OK** to close the Prerequisites dialog box.

e.      Click **OK** to close the property pages of OfficeAddInSetup.



**Figure 5: Prerequisites Dialog Box**

|  |
| --- |
| **Note:** |
| The configured prerequisite packages in your Visual Studio 2010 Setup Project are dependent on the selected build configuration. You must select the right prerequisite components for each build configuration that you use. The samples and procedures only use the default **Release** build configuration. |

Microsoft Office locates the application-level add-in by using registry keys specific to each Office Application. Keys in the HKEY\_CURRENT\_USER hive will register the add-in for each individual user of the machine whereas keys under the HKEY\_LOCAL\_MACHINE hive will register the add-in for all users of the machine.

For more information about registry keys, see [Registry Entries for Application-Level Add-Ins](http://msdn.microsoft.com/en-us/library/bb386106.aspx).

To configure the registry

1. In the **Solution Explorer**, right-click **OfficeAddInSetup**.
2. Expand **View**.
3. Click **Registry** to open the registry editor window.
4. In the **Registry(ExcelAddInSetup)** editor, expand **HKEY\_LOCAL\_MACHINE** and then **Software**.
5. Delete the **[Manufacturer]** key found under **HKEY\_LOCAL\_MACHINE\Software**. This key is automatically added when the setup project is created and is not used by the add-in.
6. Expand **HKEY\_CURRENT\_USER** and then **Software**.
7. Delete the **[Manufacturer]** key found under **HKEY\_CURRENT\_USER\Software**. This key is also added automatically and isn’t used by the add-in.
8. To add registry keys for the add-in installation right-click the **User/Machine Hive** key, select **New** and then **Key**. Use the text **Software** for the name of the new key. Right-click on the newly created Software  
   key and create a new key with the text **Microsoft.**
9. Use a similar process to create the entire key hierarchy required for the add-in registration. The following  
   key hierarchy is used for the ExcelAddIn add-in.

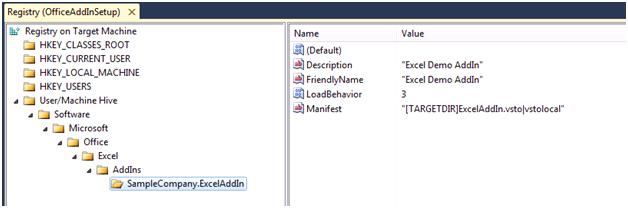
User/Machine

Hive\Software\Microsoft\Office\Excel\Addins\SampleCompany.ExcelAddIn

|  |
| --- |
| **Note:** |
| The name of your company is used as a prefix for the name of the add-in. This provide uniqueness for this part of the registry key and allows add-ins with similar names originating from different suppliers to work without accidentally overwriting each other's registration keys. This does not provide full uniqueness but should suffice to prevent collisions. |

|  |  |  |
| --- | --- | --- |
| 10. |  | Right-click the **SampleCompany.ExcelAddIn** key, select **New** and click **String value**. Use the text **Description** for the new value. |
| 11. |  | Use this step to add three more values. Use the following names and data type. |
|  | a. | **FriendlyName** of type **String** |
|  | b. | **LoadBehavior** of type **DWORD** |
|  | c. | **Manifest** of type **String** |
| 12. |  | Right-click the **Description** value in the registry editor and click **Properties Window**. In the **Properties Window** enter **Excel Demo AddIn** for the **Value** property. |
| 13. |  | Select the **FriendlyName** key in the registry editor. In the **Properties Window**, change the **Value** property to **Excel Demo AddIn**. |
| 14. |  | Select the **LoadBehavior** key in the registry editor. In the Properties Window, change the **Value** property to **3.** |
|  |  | The value 3 for the LoadBehavior value indicates that the add-in should be loaded at startup of the host application. For more information about load behavior, see  [Registry Entries for Application-Level Add-Ins](http://msdn.microsoft.com/en-us/library/bb386106.aspx). |

|  |  |  |
| --- | --- | --- |
| 15. |  | Select the **Manifest** key in the registry editor. In the **Properties Window**, change the **Value** property to **[TARGETDIR]ExcelAddIn.vsto|vstolocal** |

****

**Figure 6: Setting up registry keys**

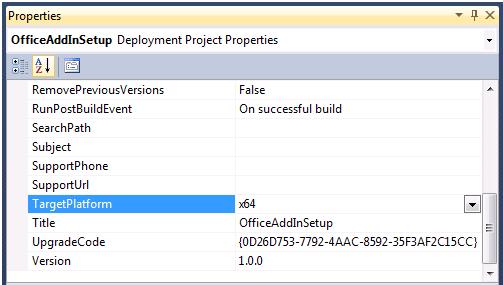
|  |
| --- |
| **Note:** |
| There are a few interesting notes on how the Visual Studio Tools for Office runtime locates the add-in. Visual Studio Tools for Office runtime searches for the deployment manifest indicated by the Manifest value. The [TARGETDIR] part is a macro that the Windows Installer expands to the actual folder where the add-in is installed. This macro expands to include the trailing \ character, so the filename of the deployment manifest is appended as ExcelAddIn.vsto without the \ character. Finally there is the use of the **vstolocal** postfix. This tells the Visual Studio Tools for Office runtime that the add-in should load from the location indicated by the Manifest value, and not loaded into the Clickonce cache. Removing this postfix will cause the runtime to copy the customization into the ClickOnce cache. |

|  |
| --- |
| **Warning:** |
| You should be very careful with the Registry Editor in Visual Studio. For example, if you accidentally set DeleteAtUninstall for the wrong key, you might delete an active part of the registry, leaving the user computer in an inconsistent, or even worse, broken state. |

Special considerations for Registering All User Add-ins for Office 2010 64-bit: Office 2010 64 bit looks for the add-in registry keys under the 64-bit hive. In order to register add-ins under the native 64-bit registry hive, the setup project’s target platform must be set to 64-bit only.

* Select the **OfficeAddInSetup** project in solution explorer.
* Go to **Properties** window and set **TargetPlatform** property to **x64**.

You cannot have a single MSI that will install for 32-bit as well as 64-bit Office versions. You must create two separate MSI packages that target the 32-bit and 64-bit versions of Office separately.



**Figure 7: Target Platform for registering add-ins with 64-bit Office 2010**

When the user does not execute the setup bootstrapper (setup.exe), it is possible that the MSI package will install on a computer without the required prerequisites. Launch conditions are used to prevent the ExcelAddIn from installing on computers without installed prerequisites.

To see if the Visual Studio 2010 Tools for Office runtime is installed, check the following registry keys on the user computer.

To configure launch conditions for Visual Studio 2010 Tools for Office Runtime

1. In **Solution Explorer**, right-click **OfficeAddInSetup**.
2. Expand **View**.
3. Click **LaunchConditions**.
4. In the **Launch Conditions(OfficeAddInSetup)** editor, right-click **Requirements on Target Machine**,  
   and then click **Add Registry Launch Condition**. This search condition searches the registry for a  
   key the Visual Studio Tools for Office runtime installs. The value of the key is then available to the various pieces  
   of the installer through a named property. The launch condition uses the property defined by the search condition to check for a certain value.
5. In the **Launch Conditions(OfficeAddInSetup)** editor, select the **Search for RegistryEntry1** search condition, right-click the condition and select **Properties Window**.
6. In the **Properties** window, perform the following tasks.

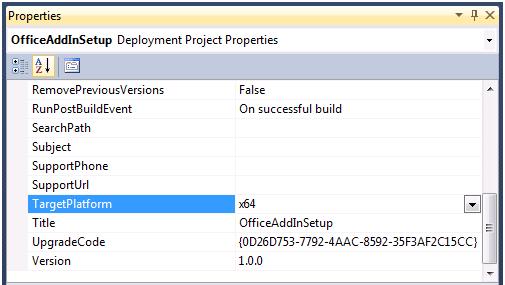
a.     Set the value of **(Name) to Search for VSTO 2010 Runtime**.

b.     Change the value of **Property** to **VSTORuntimeRedist**.

c.     Set the value of **RegKey** to **SOFTWARE\Microsoft\VSTO Runtime Setup\v4R**

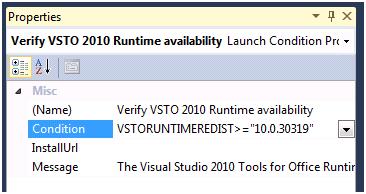
d.     Leave the **Root** property set to **vsdrrHKLM**.

e.     Change the **Value** property to **Version**.



**Figure 8: Properties Window for the Search for Runtime Launch Condition**

1. In the **Launch Conditions(OfficeAddInSetup)** editor, select the **Condition1** launch condition, right-click the condition and select **Properties Window**.
2. In the Properties window, perform the following tasks.  
   a.     Set **(Name)** to **Verify VSTO 2010 Runtime availability**.  
     
   b.     Change the value of the **Condition**property to the following  
   **VSTORUNTIMEREDIST>="10.0.30319"**  
     
   c.     Leave the **InstallURL** property blank.  
     
   d.Change the value of the **Message** property to **The Visual Studio 2010 Tools for Office  
   Runtime is not installed. Please run Setup.exe**.



**Figure 9: Properties Window for the Verify Runtime Availability launch condition**

The Visual Studio 2010 Tools for Office runtime also ships with Microsoft Office 2010. However at the time of Office 2010 RTM, the runtime with Office only supports Office solutions that target the .NET Framework 3.5. If your solution targets the .NET Framework 3.5, it can run either if Office 2010 is installed or if the Visual Studio 2010 Tools for Office Runtime redistributable is installed. If your Office solutions target the .NET Framework 4, you must redistribute the Visual Studio 2010 Tools for Office runtime. The launch condition above explicitly checks for the  
presence of the redistributable runtime. The following procedure demonstrates how you can modify the launch condition so that it will also check for the presence for Microsoft Office 2010.

To configure launch conditions for Visual Studio 2010 Tools for Office Runtime

1. In the **Launch Conditions(OfficeAddInSetup)** editor, right-click **Search Target Machine**, and then click **Add Registry Search**. Configure this additional search condition to search for the VSTO Runtime installed through Office 2010. The launch condition can use the property defined by the search condition in addition to the property for the RUNTIMEREDIST to check whether an appropriate VSTO Runtime is present on the end users system.
2. Select the **Search for RegistryEntry1** search condition, right-click the condition and select **Properties Window**.
3. In the **Properties** window, perform the following tasks.

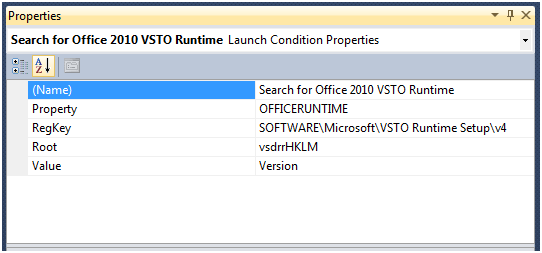
a.     Set the value of **(Name)** to Search for Office 2010 VSTO Runtime.

b.     Change the value of **Property** to **OfficeRuntime**.

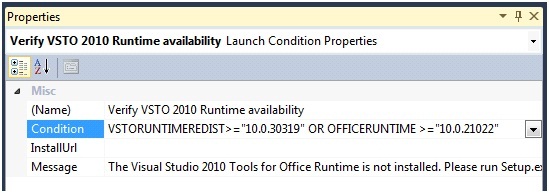
c.     Set the value of **RegKey** to **SOFTWARE\Microsoft\VSTO Runtime Setup\v4**

d.     Leave the **Root** property set to **vsdrrHKLM**.

e.     Change the **Value** property to **Version**.

  
  
**Figure 10: Properties Window for the Search for Runtime launch condition**

1. In the **Launch Conditions(OfficeAddInSetup)** editor, select the **Verify VSTO 2010 Runtime availability** launch condition defined earlier, right-click the condition and select **Properties Window**.
2. Change the value of the **Condition** property to the following **VSTORUNTIMEREDIST>="10.0.30319" OR OFFICERUNTIME>=”10.0.21022”**

  
  
**Figure 11: Properties Windows for the Verify Runtime Availability through Redist or Office 2010 launch condition**

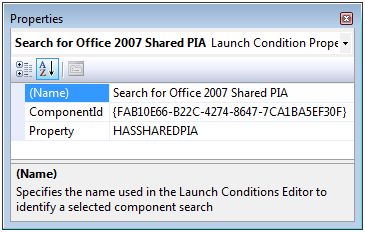
To configure launch conditions for Office 2007 Shared PIA

In the **Launch Conditions(ExcelAddInSetup)** editor, right-click **Requirements on Target Machine**, and then **click Add Windows Installer Launch Condition**. This launch condition searches for the Microsoft Office 2007 Shared Primary Interop Assembly by searching for the specific component ID.

Right-click **Search for Component1** and click **Properties** **Window** to show the properties of the launch condition.

In the **Properties** **Window**, change the values of the following properties:

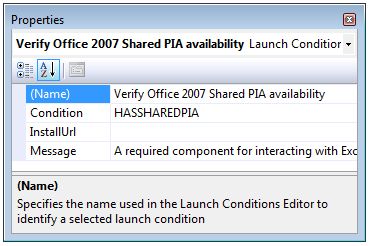
1. Set the value of the **(Name)** property to **Search for Office 2007 Shared PIA**
2. Set the value of the **ComponentID** property to **{FAB10E66-B22C-4274-8647-7CA1BA5EF30F}**.
3. Set the value of the **Property** property to **HASSHAREDPIA**.

**Figure 15. Launch Conditions Properties Window - Search for Office Shared PIA**  
  


In the **Launch Conditions(ExcelAddInSetup)** editor, right-click **Condition1** and click **Properties** **Window** to show the properties of the launch condition.

Set the properties of **Condition1** using the following information:

1. Set **(Name)** to **Verify Office 2007 Shared PIA availability**.
2. Set **Condition** to **HASSHAREDPIA**.
3. Leave **InstallUrl** blank.
4. Set **Message** to **A required component for interacting with Excel 2007 is not available. Please run setup.exe**.

**Figure 16. Launch Conditions Properties Window - Verify Office Shared PIA**  
  


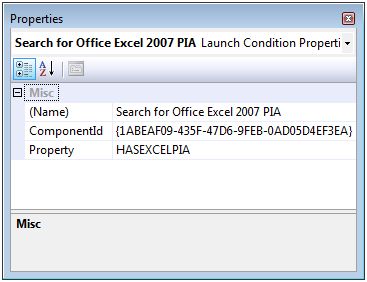
To configure launch conditions for Office EXCEl 2007 PIA

In the **Launch Conditions(ExcelAddInSetup)** editor, right-click **Requirements on Target Machine**, and then click **Add Windows Installer Launch Condition**. This launch condition searches for the Microsoft Office Excel 2007 Primary Interop Assembly by searching for the specific component ID.

Right-click **Search for Component1** and click **Properties Window** to show the properties of the launch condition.

In the **Properties** window, change the values of the following properties.

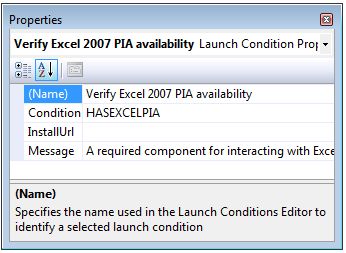
1. Set the value of the **(Name)** property to **Search for Office Excel 2007 PIA**
2. Set the value of the **ComponentID** property to **{1ABEAF09-435F-47D6-9FEB-0AD05D4EF3EA}**.
3. Set the value of the **Property** property to **HASEXCELPIA**.

**Figure 13. Launch Conditions Properties Window - Search for Excel PIA**  
  


In the **Launch Conditions(ExcelAddInSetup)** editor, right-click **Condition1** and click **Properties** **Window** to show the properties of the launch condition.

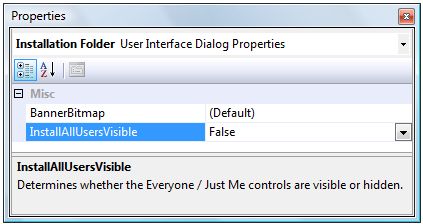
Set the properties of **Condition1** using the following information:

1. Set **(Name)** to **Verify Excel 2007 PIA availability**.
2. Set **Condition** to **HASEXCELPIA**.
3. Leave **InstallUrl** blank.
4. Set **Message** to **A required component for interacting with Excel 2007 is not available. Please run setup.exe**.

**Figure 14. Launch Conditions Properties Window - Verify PIA availability**  
  


To configure a single-user installation

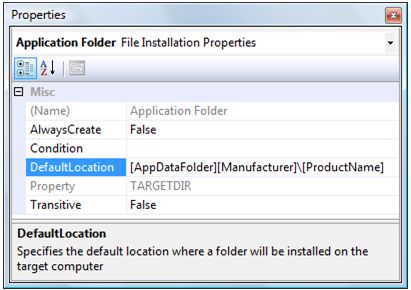
1. In the **Solution Explorer**, right-click the **ExcelAddInSetup** project, expand **View** and select **User** **Interface**.
2. In the **User Interface (ExcelAddInSetup)** editor, right-click **Installation** **Folder** and select **Properties** **Window**.
3. In the **Properties** window, set the value of the **InstallAllUsersVisible** property to **False**.

**Figure 18. Single User Installation**  
  


To allow non-administrative users to install the Visual Studio Tools for Office solution, you need to verify that no files are copied to folders that are restricted to the user. The application data folder for the user is a suitable place to install the solution.

To change the default installation folder

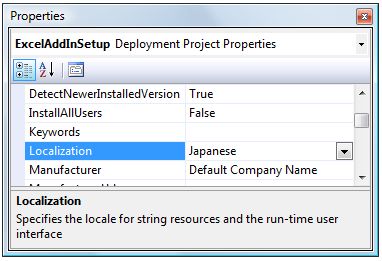
1. In **Solution Explorer**, right-click the Setup project, expand **View** and click **File System**.
2. In the **File System Editor**, right-click the **Application Folder** node and click **Properties** **Window**.
3. In the **Properties** window, set **DefaultLocation** to **[AppDataFolder][Manufacturer]\[ProductName]**. Note the slash between the manufacturer and product name.

**Figure 19. Default Installation Folder**  
  


If you downloaded a language pack for the Visual Studio Tools for Office runtime you should select the appropriate language in the setup project properties. Selecting the language for the setup project provides localized messages during the setup process as well as for the Visual Studio Tools for Office runtime.

To deploy a language pack

1. In the **Solution Explorer**, click the **ExcelAddInSetup** project.
2. In the **Properties** window, for the **Localization** property, select the language that you want to use. Note that this should be a language that you have already downloaded to the appropriate location in the bootstrapper directory.

**Figure 20. Language Pack Properties**  
  


The setup project is not compiled as part of a general solution compilation because the build can take a long time on larger projects. You need to manually compile the setup project and copy the resulting files to an accessible location.

To build the setup project

1. In the **Solution Explorer**, right-click the **ExcelAddInSetup** project and click **Build**.
2. Using **Windows** **Explorer**, navigate to the output directory of the **ExcelAddInSetup** project found in the VSTO v3 Deployment Demo\ExcelAddInSetup\Release folder. Copy all files to a location that users can access. Sample locations include a network share, CD or USB stick.

It is a good idea to test the basic setup before you add more advanced features to the setup project. You should perform the following steps on a test computer where the add-in is not installed.

To test the ExcelAddIn setup

1. Navigate to the location where you copied **ExcelAddInSetup** in the previous procedure.
2. Right-click the setup.exe file and click **Open** to install the **ExcelAddInSetup** add-in. Accept any Software License Terms that appear, and complete the setup wizard to install the add-in on the user computer.

When setup.exe is executed, the required prerequisite packages are not available yet on the target computer. These prerequisites are installed before the MSI with the ExcelAddIn customization is installed. The verification of prerequisite components using launch conditions is part of the MSI. Hence the configured launch conditions are checked after the installation of prerequisite components. This allows you to prepare the target computer before the MSI is executed.

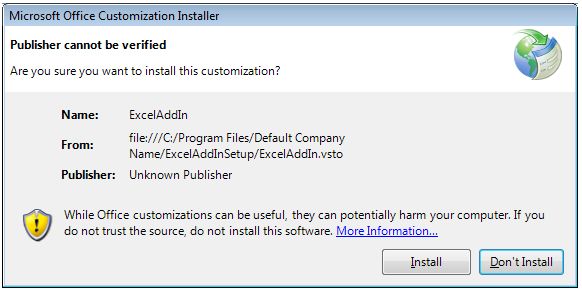
Because the user can bypass the installation of prerequisite components by not executing setup.exe but by executing the MSI directly, the MSI checks the target computer using the launch conditions. One visible side-effect is that you might be presented with a user interface pertaining to the installation of the primary interop assemblies for the 2007 Microsoft Office system before the user interface for the installation of your Visual Studio Tools for Office solution is displayed.

You can create the trust relationship as part of the installation process because running the setup project is considered a trust decision.

Perform the following procedure on a separate test computer.

To test the ExcelAddIn

1. Open Excel. You are presented with a trust prompt asking whether it is safe to execute the add-in.
2. Click **Install** in the **Microsoft Office Customization Installer** dialog box to trust and run the add-in. The following section uses other methods to define the trust relationship between Excel and your add-in.

**Figure 21. Microsoft Office Customization Installer**  
  


|  |
| --- |
| **Cc563937.note(en-us,office.12).gifNote:** |
| This dialog box is somewhat misleading for Visual Studio Tools for Office add-ins that can be locally installed using Windows Installer. The add-in is actually already installed, but it is not trusted to run yet by the user. For locally installed add-ins the Install button does not install the add-in, but it creates the trust-relationship to allow the add-in to execute. |

1. The add-in is executed.

### Table 3. Component IDs of the Redistributable Primary Interop Assemblies for the Microsoft Office System

|  |  |
| --- | --- |
| **Primary interop assembly** | **Office 2007 component ID** |
| Excel | {1ABEAF09-435F-47D6-9FEB-0AD05D4EF3EA} |
| InfoPath | {F1B5AE30-CB00-4DCF-978B-07D33B034ADB} |
| Outlook | {ED569DB3-58C4-4463-971F-4AAABB6440BD} |
| PowerPoint | {04E73476-518E-4B6A-8E10-021A00078847} |
| Visio | {C1F1028F-D91A-43E8-A117-4F7CAFD7A041} |
| Word | {816D4DFD-FF7B-4C16-8943-EEB07DF989CB} |
| Microsoft Forms 2.0 | {835AC3CE-E36B-4D65-B50F-2863A682ABEE} |
| Microsoft Graph | {580CB155-841D-4D48-9F59-866A035C2241} |
| Smart Tag | {00B41853-4377-4AD8-AD44-8404E0D331EC} |
| Office Shared | {FAB10E66-B22C-4274-8647-7CA1BA5EF30F} |
| Project | {957A4EC0-E67B-4E86-A383-6AF7270B216A} |

**Component IDs of the Redistributable Primary Interop Assemblies for Microsoft Office 2010**

|  |  |
| --- | --- |
| **Primary interop assembly** | **Office 2010 component ID** |
| Excel | {EA7564AC-C67D-4868-BE5C-26E4FC2223FF} |
| InfoPath | {4153F732-D670-4E44-8AB7-500F2B576BDA} |
| Outlook | {1D844339-3DAE-413E-BC13-62D6A52816B2} |
| PowerPoint | {EECBA6B8-3A62-44AD-99EB-8666265466F9} |
| Visio | {3EA123B5-6316-452E-9D51-A489E06E2347} |
| Word | {8B74A499-37F8-4DEA-B5A0-D72FC501CEFA} |
| Microsoft Forms 2.0 | {B2279272-3FD2-434D-B94E-E4E0F8561AC4} |
| Microsoft Graph | {011B9112-EBB1-4A6C-86CB-C2FDC9EA7B0E} |
| Smart Tag | {7102C98C-EF47-4F04-A227-FE33650BF954} |
| Office Shared | {64E2917E-AA13-4CA4-BFFE-EA6EDA3AFCB4} |
| Project | {957A4EC0-E67B-4E86-A383-6AF7270B216A} |