

Introduction to WiX

What is WiX?

The Windows Installer XML (WiX) platform is a set of tools and specifications that allow you to easily create Windows Installer database files (MSI and MSM). The WiX tools model the traditional compile and link model used to create executables from source code. For WiX, source code is written in xml files. These files are validated against a schema, wix.xsd, then processed by a preprocessor, compiler, and linker to create the desired result. The WiX platform has been designed to allow for the easy creation of multiple Windows Installer databases from a small set of source files.

[**Schema**](#)

[Overview](#)

[Authoring](#)

[Tools](#)

[WiX Files](#)

[Building WiX](#)

[Blogs](#)

[Getting Help](#)

Windows Installer XML Overview

Introduction

Windows Installer XML, or WiX, provides a schema that describes a Windows Installer database (MSI or MSM), as well as tools to convert the XML description files into a usable database. The second version of the schema, `wix.xsd`, adds extra content to ease the creation of multiple Windows Installer databases from a single set of XML documents. The WiX tools model the traditional compile and link model used to create executables from source code. This document provides a brief introduction how to use the tools to compile and link WiX source code into Windows Installer databases.

Note: This document assumes you have a working knowledge of the Windows Installer database format.

.wxs & .wixobj – Windows Installer Xml Files

A .wxs file is the extension used by all source files in the Windows Installer XML system. These .wxs files are analogous to .cpp files for C++ or .cs files for C#. The .wxs files are preprocessed then compiled into WiX object files which use the extension .wixobj. When all of the source files have been compiled into object files, the linker is used to collect the object files together and create a Windows Installer database. More details on the compiler and linker are provided later in this document.

Structure of .wxs files

All .wxs files are well-formed XML documents that contain a single root element named `<Wix/>`. The rest of the source file may or may not adhere to the WiX schema before preprocessing. However, after being preprocessed all source files must conform to the WiX schema or they will fail to compile.

The root `<Wix/>` element can contain at most one of the following two elements as children: `<Product/>`, `<Module/>`. However, there can be an unbounded number `<Fragment/>` elements as children of the root `<Wix/>` element. When a source file is compiled into an object file, each instance of these elements creates a new section in the object file. Therefore, these three elements are often referred to as section elements.

It is important to note, that there can be only one `<Product/>` or `<Module/>` section element per source file because they are compiled into special sections called entry sections. Entry sections are used as starting points in the linking process. Sections, entry sections, and the entire linking process are described in greater detail later in this document.

The children of the section elements define the contents of the Windows Installer database. You'll recognize `<Property/>` elements that map to entries in the Property table and a hierarchy of `<Directory/>` elements that build up the Directory table. Most elements contain an "Id" attribute that will act as the primary key for the resulting row in the Windows Installer database. Note, in the first release of the WiX schema the primary key was represented by the text of the element. This location for the primary key was undesirable for several reasons and has been moved to the "Id" attribute. In most cases, the "Id" attribute also defines a symbol when the source file is compiled into an object file.

Symbols and references

Every symbol in an object file is composed of the element name plus the unique identifier from the “Id” attribute. Symbols are important because they can be referenced by other sections from any source file. For example, a <Directory/> structure can be defined in a <Fragment/> in one source file and a <Component/> can be defined under a different source file’s <Fragment/>. By making the <DirectoryRef/> element a parent of the <Component/> an explicit reference is created that references the symbol defined by a <Directory/> in the first source file. The linker is then responsible for stitching the symbol and the reference together in a single Windows Installer database. In some cases, implicit references are generated by the compiler while processing a source file. These implicit references behave identically to explicit references.

In addition to the simple references described above, WiX supports specific complex references. Complex references are used in cases where the linker must generate extra information to link the symbol and reference together. The perfect example of a complex reference is in the Windows Installer’s Feature/Component relationship. When a <Component/> is referenced explicitly by a <Feature/> through a <ComponentRef/> element, the linker must take the <Feature/>’s symbol and the <Component/>’s symbol and add an entry to the FeatureComponents table.

This Feature/Component relationship is even more complex because certain elements in a <Component/>, for example <Shortcut/>, have references back to the primary Feature associated with the Component. These references from a child element of a <Component/> are called reverse references or sometimes feature backlinks. Processing complex references and reverse references is probably the most difficult work the linker has to do.

Note the process of defining and referencing symbols is new to the second version of the WiX toolset. Previously, it was necessary to package Components into Merge Modules and use the merge process to do rudimentary symbol linking. This new system for defining symbols is more flexible, and avoids the overhead of ensuring each Merge Module’s

tokens are unique.

Structure of the .wixobj file

A .wixobj file is created by the compiler for each source file compiled. The .wixobj file is an XML document that follows the objects.xsd schema defined in the WiX project. As stated above the .wixobj file contains one or more sections that, in turn, contain symbols and references to other symbols.

While the symbols and references are arguably the most important pieces of data in the .wixobj file, they are rarely the bulk of the information. Instead, the majority of most .wixobj files are composed of <table/>, <row/> and <field/> elements that provide the raw data to be placed in the Windows Installer database. In many cases, the linker will not only process the symbols and references but also use and update the raw data from the .wixobj file.

It is interesting to note that the object file schema, objects.xsd, uses camel casing where the source file schema, wix.xsd, uses Pascal casing. This was a conscious choice to indicate that the object files are not intended to be edited by the user. In fact, all schemas that defines data to be processed only by the WiX tools use camel casing.

candle – Windows Installer XML Compiler

Windows Installer XML compiler is exposed by candle.exe. candle is responsible for preprocessing the input .wxs files into valid well-formed XML documents against the WiX schema, wix.xsd. Then, each post-processed source file is compiled into a .wixobj file.

The compilation process is relatively straight forward. The WiX schema lends itself to a simple recursive descent parser. The compiler processes each element in turn creating new symbols, calculating the necessary references and generating the raw data for the .wixobj file.

The second version of candle is not significantly different from the first implementation. Any changes were either made to enable the new symbol/reference linking or based on feedback from customers. Some of the differences between versions include: the new object file format is XML instead of MSI, modularization of primary keys now happens at link time, and binary streams are imported at link time.

light – Windows Installer XML Linker

The Windows Installer XML linker is exposed by light.exe. light is responsible for processing one or more .wixobj files, retrieving metadata from various external files and creating a Windows Installer database (MSI or MSM). When necessary, light will also create cabinets and embed streams in the created Windows Installer database.

The linker begins by searching the set of object files provided on the command line to find the entry section. If more than one entry section is found, light fails with an error. This failure is necessary because the entry section defines what type of Windows Installer database is being created, a MSI (<Product/>) or MSM (<Module/>). It is not possible to create two databases from a single link operation.

While the linker was determining the entry section, the symbols defined in each object file are stored in a symbol table. After the entry section is found, the linker attempts to resolve all of the references in the section by finding symbols in the symbol table. When a symbol is found in a different section, the linker recursively attempts to resolve references in the new section. This process of gathering the sections necessary to resolve all of the references continues until all references are satisfied. If a symbol cannot be found in any of the provided object files, the linker aborts processing with an error indicating the undefined symbol.

After all of the sections have been found, complex and reverse references are processed. This processing is where Components and Merge Modules are hooked to their parent Features or, in the case of Merge Modules, Components are added to the ModuleComponents table. The reverse reference processing adds the appropriate Feature identifier to the necessary fields for elements like, Shortcut, Class, and TypeLib.

Once all of the references are resolved, the linker processes all of the rows retrieving the language, version, and hash for referenced files, calculating the media layout, and including the necessary standard actions to ensure a successful installation sequence. This part of the processing typically ends up generating additional rows that get added

associated with the entry section to ensure they are included in the final Windows Installer database.

Finally, light works through the mechanics of generating IDT files and importing them into the Windows Installer database. After the database is fully created, the final post processing is done to merge in any Merge Modules and create a cabinet if necessary. The result is a fully functional Windows Installer database.

Authoring

Authoring is the process of writing the WiX source files necessary to create a Windows Installer database. The source files, which usually have the extension .wxs, are XML documents that must conform to the WiX schema (found in wix.xsd).

[Getting Started](#)

[WiX Standard CustomActions](#)

[WiX Online Tutorial](#)

[Extensions](#)

[Patch Building](#)

[Using the WixUI dialog library](#)

[WiX Schema Reference](#)

[PubCA Schema Reference](#)

Getting Started

WiX can create Windows Installer databases which include: Windows Installer packages, MSI files, and Merge Modules, MSM files. We'll start by creating a Windows Installer package so that you'll have something that you can install and uninstall quickly. Then, we'll create a Merge Module and merge it into our example Windows Installer package. Finally, I'll cover a few more advanced topics such as how to define CustomActions and a deeper look into symbols and references.

Topics:

1. [Your first .wxs file](#)
2. [Creating Merge Modules](#)
3. [Adding Custom Actions](#)
4. [Msi Tables to WiX Schema Translation Guide](#)

Authoring Your First .wxs File

Pick your favorite XML editor—for all of the examples, I'll use notepad-- and create a new file called “product.wxs”. Nothing about that name is special, but the .wxs extension lets us know that this is a Windows Installer Xml Source File. Now, let's add the three lines of text all .wxs files have:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
</Wix>
```

That forms the outer skeleton for our source file and, honestly, any other source file we ever want to get compiled. You can feed this empty source file to candle.exe and get out an empty object file. Tell you what, let's do that. Follow the following steps and you should see very similar output:

```
C:\test> candle product.wxs
Microsoft (R) Windows Installer Xml Compiler version 1.0.1220.15022
Copyright (C) Microsoft Corporation 2003. All rights reserved

C:\test> type product.wixobj
<?xml version="1.0" encoding="utf-8"?><wixObject
xmlns="http://schemas.microsoft.com/wix/2003/04/objects"
src="C:\test\product.wxs" />

C:\test>
```

Let's notice a couple things before continuing. First, notice that when there is no error candle doesn't print any text other than its header. In fact, you can even suppress the header output by specifying "-nologo" on the command line. In that case, candle will print nothing unless there is a failure. Second, notice that the path to the original source file is stored in the .wixobj file. This can be useful when tracking down where an error is coming from. In fact, the linker uses that "src" attribute to print more informative error messages when it encounters a problem.

Okay, now that we've seen an empty source file create an empty object file, let's create an installable Windows Installer package. Add the following content to your product.wxs file:

```

<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='12345678-1234-1234-1234-123456789012' Name='Test Pa
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
    <Package Id='12345678-1234-1234-1234-123456789012'
      Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Win
      Manufacturer='Microsoft Corporation' InstallerVersio

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Component Id='MyComponent' Guid='12345678-1234-1234-1234-
    </Directory>

    <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
      <ComponentRef Id='MyComponent' />
    </Feature>
  </Product>
</Wix>

```

This should allow us to create a MSI with a ProductCode of "{12345678-1234-1234-1234-123456789012}" with ProductLanguage of "1033" and a ProductVersion of "1.0.0.0". All of that information is taken from the <Product/> element. The <Package/> element defines all of the information that goes in our MSI's summary information stream. Finally, a simple <Directory/> and <Feature/> tree is created with a single <Component/>. This is enough to get our MSI registered on the machine.

So let's compile, link, and install then take a look at the registered packages for our MSI. Follow the instructions:

Note: This MSI requires admin privileges and will silently fail if you are not installing as an Administrator.

```

C:\test> candle product.wxs
Microsoft (R) Windows Installer Xml Compiler version 1.0.1220.15022
Copyright (C) Microsoft Corporation 2003. All rights reserved

product.wxs

C:\test> light product.wixobj
Microsoft (R) Windows Installer Xml Linker version 1.0.1220.15022
Copyright (C) Microsoft Corporation 2003. All rights reserved

C:\test> msiexec /i product.msi

```

```
C:\test> \\delivery\tools\msiconfig.exe
.
.
.
{12345678-1234-1234-1234-123456789012} Test Package
.
.
.
```

You should see your "Test Package" listed with all the other Windows Installer packages installed on your machine. You can also go to Add/Remove Programs in the Control Panel and see "Test Package" registered there. Go ahead and remove the package now, so we don't forget it later.

Great! Now that we have a package that installs and uninstalls properly, let's actually install something. So, create a new text file called "readme.txt" next to your "product.wxs" file and type a message to yourself in there. "Hello, World!" is a favorite. Then, we need to modify the product.wxs to tell it about the file:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='12345678-1234-1234-1234-123456789012' Name='Test Pa
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
    <Package Id='12345678-1234-1234-1234-123456789012'
      Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Wi
      Manufacturer='Microsoft Corporation' InstallerVersi

    <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDir' Name='TestProg' LongName='Test Pr
          <Component Id='MyComponent' Guid='12345678-1234-1234
            <File Id='readme' Name='readme.txt' DiskId='1' sr
          </Component>
        </Directory>
      </Directory>
    </Directory>
  </Directory>

  <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
    <ComponentRef Id='MyComponent' />
  </Feature>
</Product>
</Wix>
```

You should be able to compile, link, and install that MSI and see that you do get a directory called "Test Program" in your system's "Program Files" folder. In that "Test Program" directory should be the "readme.txt" file you created with the message to yourself. Spiffy, eh? Again, remember to uninstall the MSI so you can rebuild and install it again later.

Believe it or not, that's all there is to creating a Windows Installer package. Sure, you can add UI and things like that now, but we've covered the basics. Everything just comes down to filling in the right XML elements. So, let's move on and look at creating a Merge Module we can incorporate into our spiffy new package.

Creating Merge Modules

Creating a Merge Module is very much like creating a Windows Installer package. So, let's create a new text file called "module.wxs" and put the standard skeleton in it, as so:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
</Wix>
```

Then to create a Merge Module, we add the <Module/> element and add the required attributes:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Module Id='TestModule' Guid='87654321-4321-4321-4321-2109876543'
    <Package Id='87654321-4321-4321-4321-210987654321' Descriptio
      Comments='This is my first attempt at creating a Wi
      Manufacturer='Microsoft Corporation' InstallerVersi

  </Module>
</Wix>
```

You can, if you wish, compile and link that code. You'll get a very small and not very interesting .msm file from light. So, let's add a text file to this Merge Module like we did to the Windows Installer package above. First, create a text file called "readme2.txt" and put a different message to yourself in there. Then, update the source code to include the new file:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Module Id='TestModule' Guid='87654321-4321-4321-4321-2109876543'
    <Package Id='87654321-4321-4321-4321-210987654321' Descriptio
      Comments='This is my first attempt at creating a Wi
      Manufacturer='Microsoft Corporation' InstallerVersi

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='MyModuleDirectory' Name='.'>
        <Component Id='MyModuleComponent' Guid='87654321-4321-4
          <File Id='readme2' Name='readme2.txt' src='readme2.t
        </Component>
      </Directory>
    </Directory>
  </Module>
```

```
</Wix>
```

That's it! You now have a Merge Module that can be shared with other teams to install your "readme2.txt" file. Now that we have a Merge Module, let's actually use it in a Windows Installer package.

Incorporating a Merge Module into a .wxs file

Merge Modules can only be merged into Windows Installer package. Fortunately, we have a .wxs file that creates a Windows Installer package from our first experiments with WiX. So, let's add the two lines (yes, only two lines are necessary) to merge in your new Module. Open your "product.wxs" source file again, and add:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='12345678-1234-1234-1234-123456789012' Name='Test Pa
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'
    <Package Id='12345678-1234-1234-1234-123456789012' Descriptio
      Comments='This is my first attempt at creating a Win
      Manufacturer='Microsoft Corporation' InstallerVersio

    <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDir' Name='TestProg' LongName='Test Pr
          <Component Id='MyComponent' Guid='12345678-1234-1234
            <File Id='readme' Name='readme.txt' DiskId='1' sr
              </Component>

          <Merge Id='MyModule' Language='1033' src='module.msm
            </Directory>
          </Directory>
        </Directory>

        <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
          <ComponentRef Id='MyComponent' />
          <MergeRef Id='MyModule' />
        </Feature>
      </Product>
    </Wix>
```

Now when you compile your Windows Installer package source file, it will include the installation logic and files from the Merge Module. The next time you install the "product.msi", you should see two text files in the "Test Program" directory instead of one.

Adding Custom Actions

Now that you're comfortable with the basics for creating Windows Installer packages, let's take it to the next level and add a CustomAction. Since every release of the Windows Installer XML toolset comes with a drop of the WiX Server CustomActions, we'll use those for our example. So go now to the `wix\bin\ca` directory and copy the "`sca*.dll`" to the same directory as your "`product.wxs`" and "`readme.txt`" files. You should have "`scasched.dll`" and "`scaexec.dll`".

Rather than put the CustomAction definitions in the same source file as our product definition, let's exercise a little modularity and create a new source file to define the CustomActions called "`sca.wxs`". Let's begin by adding the immediate CustomAction that reads the custom server tables and schedules the deferred actions.

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Fragment Id="ServerCustomActions">
    <CustomAction Id='ConfigureIIs' BinaryKey='ScaSchedule' DllEn
      Return='check' />
    <CustomAction Id='ConfigureSql' BinaryKey='ScaSchedule' DllEn
      Return='check' />

    <Binary Id='ScaSchedule' src='scasched.dll' />
  </Fragment>
</Wix>
```

That little bit of code should compile but it will not link. Remember linking requires that you have an entry section and a `<Fragment/>` alone is not an entry section. We would need to link this source file along with a source file that contained `<Product/>` or `<Module/>` to successfully complete. Before we bother getting everything to link properly, let's add the deferred CustomActions to this source file since they are as important as the immediate CustomActions you already added.

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Fragment Id="ServerCustomActions">
    <CustomAction Id='ConfigureIIs' BinaryKey='ScaSchedule' DllEn
      Return='check' />
    <CustomAction Id='ConfigureSql' BinaryKey='ScaSchedule' DllEn
```

```

        Return='check' />

    <CustomAction Id='ErrorOut' BinaryKey='ScaExecute' DllEntry='
        Return='check' />

    <CustomAction Id='StartMetabaseTransaction' BinaryKey='ScaExe
        DllEntry='StartMetabaseTransaction' Execute='de
    <CustomAction Id='RollbackMetabaseTransaction' BinaryKey='Sca
        DllEntry='RollbackMetabaseTransaction' Execute=
    <CustomAction Id='CommitMetabaseTransaction' BinaryKey='ScaEx
        DllEntry='CommitMetabaseTransaction' Execute='c

    <CustomAction Id='CreateMetabaseKey' BinaryKey='ScaExecute'
        DllEntry='CreateMetabaseKey' Execute='deferred'
    <CustomAction Id='DeleteMetabaseKey' BinaryKey='ScaExecute'
        DllEntry='DeleteMetabaseKey' Execute='deferred'
    <CustomAction Id='CreateAspApp' BinaryKey='ScaExecute'
        DllEntry='CreateAspApp' Execute='deferred' Retu
    <CustomAction Id='WriteMetabaseValue' BinaryKey='ScaExecute'
        DllEntry='WriteMetabaseValue' Execute='deferred
    <CustomAction Id='WriteMetabaseMultiString' BinaryKey='ScaExe
        DllEntry='WriteMetabaseMultiString' Execute='de
    <CustomAction Id='DeleteMetabaseMultiString' BinaryKey='ScaEx
        DllEntry='DeleteMetabaseMultiString' Execute='d

    <CustomAction Id='CreateDatabase' BinaryKey='ScaExecute'
        DllEntry='CreateDatabase' Execute='deferred' Re
    <CustomAction Id='DropDatabase' BinaryKey='ScaExecute'
        DllEntry='DropDatabase' Execute='deferred' Retu
    <CustomAction Id='ExecuteSqlStrings' BinaryKey='ScaExecute'
        DllEntry='ExecuteSqlStrings' Execute='deferred'
    <CustomAction Id='RollbackExecuteSqlStrings' BinaryKey='ScaEx
        DllEntry='ExecuteSqlStrings' Execute='rollback'

    <Binary Id='ScaSchedule' src='scasched.dll' />
    <Binary Id='ScaExecute' src='scaexec.dll' />
</Fragment>
</Wix>

```

Okay, that's it. We're done with editing the "sca.wxs" source file. You have successfully defined all of the entry points into the WiX Server CustomActions. Now, how about we add a call to the WiX Server CustomActions to the example product.wxs source file you've been working with so far. Instead of configuring IIS or SQL Server (and requiring you to have one of them installed), let's just add a call to the CustomAction I use to inject errors into the installation process for testing purposes. That's the "ErrorOut" CustomAction.

```

<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='12345678-1234-1234-1234-123456789012' Name='Test Pa
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
    <Package Id='12345678-1234-1234-1234-123456789012'
      Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Window
      Manufacturer='Microsoft Corporation' InstallerVersion='

    <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDir' Name='TestProg' LongName='Test Pr
          <Component Id='MyComponent' Guid='12345678-1234-1234
            <File Id='readme' Name='readme.txt' DiskId='1' sr
              </Component>

          <Merge Id='MyModule' Language='1033' src='module.msm
            </Directory>
          </Directory>
        </Directory>
      </Directory>

      <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
        <ComponentRef Id='MyComponent' />
        <MergeRef Id='MyModule' />
      </Feature>

      <InstallExecuteSequence>
        <Custom Action='ErrorOut' After='InstallFiles' />
      </InstallExecuteSequence>
    </Product>
  </Wix>

```

Those three lines are all you need to add to your Windows Installer package source file to call the "ErrorOut" CustomAction. Now that we have two files to link together our call to light.exe gets a little more complicated. Here are the compile, link, and installation steps.

```

C:\test> candle product.wxs module.wxs sca.wxs
Microsoft (R) Windows Installer Xml Compiler version 1.0.1256.19889
Copyright (C) Microsoft Corporation 2003. All rights reserved.

product.wxs
module.wxs
sca.wxs

```

```
C:\test> light module.wixobj  
Microsoft (R) Windows Installer Xml Linker version 1.0.1256.19889  
Copyright (C) Microsoft Corporation 2003. All rights reserved.  
  
C:\test> light product.wixobj sca.wixobj -out product.msi  
Microsoft (R) Windows Installer Xml Linker version 1.0.1220.15022  
Copyright (C) Microsoft Corporation 2003. All rights reserved  
  
C:\test> msiexec /i product.msi
```

Don't be alarmed when the MSI mysteriously starts rolling back the installation. Remember after installing the files the "ErrorOut" CustomAction is called and that forces the installation to fail. MSI then rolls back the files and silently returns. Adding a success and an error dialog are exercises left to the interested reader.

Msi Tables to WiX Schema

In the WiX schema, its not always entirely obvious how the tables from the Windows Installer schema map to the WiX schema. Below are some helpful hints on how to figure out the relationships between the two schemas.

DuplicateFile Table

This is authored using a [CopyFile](#) node nested under a File node. You only need to set the Id, DestinationFolder, and DestinationName attributes.

LaunchCondition Table

This is authored using a [Condition](#) node authored under Fragment or Product. You only need to set the Message attribute.

LockPermissions Table

This is authored using [Permission](#).

MoveFile Table

This is authored using a [CopyFile](#) node nested under a Component node. You will need to set all attributes except Delete. Set Delete to 'yes' in order to use the msidbMoveFileOptionsMove option.

PublishComponent Table

The PublishComponent functionality is available in WiX by using a [Category](#). Here is a small sample of what a PublishComponent record would look like in MSI, then in WiX notation.

MSI

ComponentId	Qualifier	Component_	AppData	Feature_
{11111111-2222-3333-4444-555555555555}	1033	MyComponent	Random Data	MyFeature

WiX

```
<Component Id='MyComponent' Guid='87654321-4321-4321-4321-110987654321'>
  <Category Id='11111111-2222-3333-4444-555555555555' AppData
    Qualifier='1033' />
</Component>
.
.
.
<Feature Id='MyFeature' Level='1'>
  <ComponentRef Id='MyComponent' />
</Feature>
```

RemoveIniFile

This is authored using [IniFile](#). Just set the Action attribute to 'removeLine' or 'removeTag' as appropriate.

RemoveRegistry Table

This is authored using [Registry](#). Simply set the Action attribute to 'remove' or 'removeKey' (as appropriate) in order to get an entry in the RemoveRegistry table.

Windows Installer XML Online Tutorials

Gabor Deak Jahn maintains an impressive [online tutorial](#) about the [Windows Installer XML toolset](#). That tutorial is great way to ramp up on the WiX toolset if you are new or looking for answers to common authoring task.

Additional Resources

Rob Mensching has written an excellent [MSDN article](#) on using [Votive](#) to get started in WiX.

Windows Installer XML Standard CustomActions

The WiX toolset contains several CustomActions to handle configuring resources such as Internet Information Services web sites and virtual directories, SQL Server databases and scripts, user accounts, file shares, and more. These CustomActions are provided in two separate .wixlibs: sca.wixlib and wixca.wixlib. The former contain "Server CustomActions" while the latter has more general installation CustomActions. In the future, these .wixlib's may merge together but for now (for mostly historical reasons) they are separate.

sca.wixlib - Server CustomActions

Internet Information Services (IIS) CustomAction - create and configure web sites, virtual directories, web applications, etc.

SQL Server CustomAction - create databases and execute SQL scripts and statements.

User CustomAction - create and configure new users.

FileShare CustomAction - create and configure file shares (SMB).

[Performance Counter CustomAction](#) - install and uninstall performance counters.

wixca.wixlib - General CustomActions

Secure Objects CustomAction - secure (using ACLs) objects that standard LockPermission table cannot. For further information see the Extended attribut in [<Permission/>](#).

Service Configuration CustomAction - configure attributes of a Windows Service that the ServiceInstall table cannot.

[Quiet Execution CustomAction](#) - launch console executables without displaying a window.

XmlFile CustomAction - allows you to configure XML files as part of your installation package. For further information see [<XmlFile/>](#).

New CustomActions are always under development. Our goal is to one day have standard CustomActions for just about any need. Feel free to open a [Feature Request](#) if you have a CustomAction need.

Using the Server Custom Actions

The wix toolset contains a library of custom actions. The centerpiece of this library is the server custom action set. The server custom actions extend the set of resources that an MSI can install to include things such as web sites, file shares, user accounts, and many others. These custom actions properly associate these resources with components, and follow all the rules to properly install, uninstall and rollback the installation or uninstallation of these resources as part of their associated components. This document will outline their use with some examples.

This document assumes that the reader has an understanding of MSI custom action types, and has read "WiX Overview" and "Writing in WiX".

Server Custom Action building blocks

With each release of the wix toolset, the files `scasched.dll`, `scaexec.dll` and `sca.wixlib` are released. The two dll files are the custom action dlls which export the custom action entry points for all of the server custom actions. When you build an MSI that makes use of the server custom actions, they end up in the Binary table of the MSI. The `sca.wixlib` contains a system of wix fragments that you can link against to ensure that all of the proper error messages, custom action records, and binary records get linked into your final MSI.

The simplest way to incorporate the server custom actions into your MSIs is to copy the `sca.wixlib` and the two custom action dlls (`scasched.dll` and `scaexec.dll`) into a folder in your build environment. It is not important where this directory is, it is only important that the wixlib and the dlls are in the same directory. When you link your MSI using `light.exe`, you simply need to include the full path to `sca.wixlib` in the list of wixobjs and wixlibs you're linking.

Basic Example

First lets try an example that creates a user account when the MSI is installed.

```
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='PutGuidHere' Name='TestUserProduct' Language=
    <Package Id='PUT-GUID-HERE' Description='Test User P
      <Directory Id='TARGETDIR' Name='SourceDir'>
        <Component Id='TestUserProductComponent' Gui
          <User Id='TEST_USER1' Name='testName1' P
        </Component>
      </Directory>

      <Feature Id='TestUserProductFeature' Title='Test Use
        <ComponentRef Id='TestUserProductComponent' />
      </Feature>
    </Product>
  </Wix>
```

This is a simple example that will create a new user on the machine called "testName1" with the password "pa\$\$word". To build the MSI from this wix authoring first put the above code in a file (remember to replace the "PUT-GUID-HERE" attributes with real GUIDs), run 'candle.exe yourfile.wxs', and then run 'light.exe -out yourfile.msi yourfile.wixout sca.wixlib' (replacing sca.wixlib with the full path to sca.wixlib). Now use Orca to open up the resulting msi and take a look at the Error table, the CustomAction table, and the Binary table. You will notice that all of the relevant data for managing users has been "linked" into the MSI. This happened because you have done two key things. First, you made use of a <User/> element under a <Component/> element which indicates that a user is to be installed as part of the MSI package, and second, you linked with the sca.wixlib. Compiler support, along with the system of fragments that exist in the sca.wixlib ensure that only the data associated with the elements you

used in your wxs file are "linked" into the MSI.

The server custom action elements

In the previous example you learned that by using the `<User/>` element in your WiX authoring and then linking with the `sca.wixlib` that all of the relevant custom actions, error messages, and binary table rows were brought in automatically. The wix compiler contains support for automatically referencing the appropriate symbols in the `sca.wixlib` when you make use of specific elements such as `<User/>`. As stated in the introduction the server custom actions add the ability to install many new types of resources. Each of these resource types has one or more elements that allow you to install them with your MSI package. If you're using the `sca.wixlib`, the only things you need to know are the appropriate elements for the resources you want to install. Here is a listing of the different resource types that the server custom actions are able to install and the elements that control their installation:

- Web Sites - `<WebSite/>`
- Web Applications - `<WebApplication/>`
- Certificates - `<Certificate/>`
- SQL databases - `<SqlDatabase/>`
- SQL scripts - `<SqlScript/>`
- SQL strings - `<SqlString/>`
- Users - `<User/>`
- FileShares - `<FileShare/>`
- Perfmon Counter registration - `<PerfCounter/>`

By using the appropriate elements from this table in your wix authoring and by linking with `sca.wixlib`, you will ensure that you are properly using the wix server custom actions.

Performance Counter CustomActions

The PerfCounter element allows you to register your performance counters with the Windows API. There are several pieces that all work together to successfully register:

Your performance DLL - The DLL must export Open, Collect, and Close methods. See MSDN for more detail.

Performance registry values - The registry must contain keys pointing to your DLL and its Open, Collect, and Close methods. These are created using the Registry element.

Perfmon INI and H text files - These contain the text descriptions to display in the UI. See MSDN for lodctr documentation. [This MSDN documentation](#) is a good place to start. See below for samples re-purposed from MSDN.

The RegisterPerfmon custom action - You can link with sca.wixlib to ensure that the custom actions are included in your final MSI. See [server custom action documentation](#). The custom action calls (Un)LoadPerfCounterTextStrings to register your counters with Windows' Perfmon API. To invoke the custom action, you create a PerfCounter element nested within the File element for the Perfmon.INI file. The PerfCounter element contains a single attribute: Name. The Name attribute should match the name in the Registry and in the .INI file. See below for sample WIX usage of the <PerfCounter> element.

Sample WIX source fragment and PerfCounter.ini

```
<?xml version="1.0"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2003/01/wi">
  <Fragment>
    <DirectoryRef Id="BinDir">
      <Component Id="SharedNative" DiskId="1">

        <Registry Id="Shared_r1" Root="HKLM" Key="SYSTEM\CurrentCon
        <Registry Id="Shared_r2" Root="HKLM" Key="SYSTEM\CurrentCon
        <Registry Id="Shared_r3" Root="HKLM" Key="SYSTEM\CurrentCon
        <Registry Id="Shared_r4" Root="HKLM" Key="SYSTEM\CurrentCon

      <File Id="PERFDLL.DLL" Name="MYPERFDLL.DLL" LongName="MyPerf

      <File Id="PERFCOUNTERS.H" Name="PERF.H" LongName="PerfCounte
      <File Id="PERFCOUNTERS.INI" Name="PERF.INI" LongName="PerfCo
        <PerfCounter Name="MyApplication" />
      </File>

    </Component>
  </DirectoryRef>
</Fragment>
</Wix>
```

```
Sample PerfCounters.ini:
[info]
drivename=MyApplication
symbolfile=PerfCounters.h

[languages]
009=English
004=Chinese

[objects]
PERF_OBJECT_1_009_NAME=Performance object name
PERF_OBJECT_1_004_NAME=Performance object name in Chinese

[text]
OBJECT_1_009_NAME=Name of the device
OBJECT_1_009_HELP=Displays performance statistics of the device
OBJECT_1_004_NAME=Name of the device in Chinese
```

```
OBJECT_1_004_HELP=Displays performance statistics of the device in  
  
DEVICE_COUNTER_1_009_NAME=Name of first counter  
DEVICE_COUNTER_1_009_HELP=Displays the current value of the first c  
DEVICE_COUNTER_1_004_NAME=Name of the first counter in Chinese  
DEVICE_COUNTER_1_004_HELP=Displays the value of the first counter i  
  
DEVICE_COUNTER_2_009_NAME=Name of the second counter  
DEVICE_COUNTER_2_009_HELP=Displays the current rate of the second c  
DEVICE_COUNTER_2_004_NAME=Name of the second counter in Chinese  
DEVICE_COUNTER_2_004_HELP=Displays the rate of the second counter i  
  
PERF_OBJECT_1_009_NAME=Name of the third counter  
PERF_OBJECT_1_009_HELP=Displays the current rate of the third count  
PERF_OBJECT_1_004_NAME=Name of the third counter in Chinese  
PERF_OBJECT_1_004_HELP=Displays the rate of the third counter in Ch  
Sample PerfCounters.h:  
#define OBJECT_1      0  
#define DEVICE_COUNTER_1      2  
#define DEVICE_COUNTER_2      4  
#define PERF_OBJECT_1      8
```

Quiet Execution CustomAction

There is a qtexec custom action that is part of the wixca that can run arbitrary command lines.

Immediate execution

```
<Property Id="QtExecCmdLine" Value="command line to run"/>
<CustomAction Id="QtExec" BinaryKey="wixca" DllEntry="CAQuietExec"
<Binary Id="wixca" src="wixca.dll"/>
.
.
.
<InstallExecuteSequence>
  <Custom Action="QtExec" After="TheActionYouWantItAfter"/>
</InstallExecuteSequence>
```

This will result in running the command line in the immediate sequence. If the exit code of the command line is an error (not 0) then because Return is set to "check" it will cause the install to fail. You can change this value to "ignore" if you don't want it to cause an install failure (it will be logged still).

If you want to run more than one command line in the immediate sequence then you'll need schedule QtExec multiple times and set the QtExecCmdLine property (using a type 51 custom action) right before you want each of them executed.

Deferred execution

You can also run command lines in the deferred script using this tool by setting the custom action data property. If the code is running in immediate mode it will try to execute the value of the QtExecCmdLine if it is running in deferred (or rollback) mode it will try to execute the value of the custom action data. The custom action data is a property that is named the same as the custom action. Here's an example of authoring deferred command line execution:

```
<Property Id="QtExecDeferred" Value="command line to run"/>
<CustomAction Id="QtExecDeferred" BinaryKey="wixca" DllEntry="CAQui
<Binary Id="wixca" src="wixca.dll"/>
.
.
.
<InstallExecuteSequence>
  <Custom Action="QtExecDeferred" After="TheActionYouWantItAfter"
</InstallExecuteSequence>
```

Extensions

WiX has support for three classes of extensions

Introduction

Preprocessor Extensions allow clients to modify authoring files before they are processed by the compiler.

Compiler Extensions allow clients to custom compile authored XML into internal table representation before it's written to binary form.

Binder Extensions allow clients to feed the interlace image processing and data finalization.

Through these extensions one can extend WiX to support custom preprocessing, XML syntax compilation, or binding semantics for ones particular layout generation process.

Common Requirements

How to use each should start in the source code but they all have a few things in common

Implemented in same version of .NET 1.1 as the rest of WiX

Build a subclass of the appropriate extension object giving it a easily distinguishable name.

Build a schema of the appropriate syntax to provide validation checking where possible.

Build internal table definitions and register them with the compiler.

Build overrides for extendable methods and virtual members which will get invoked at the appropriate location during the single pass compile.

Build extension into a DLL.

Place extension DLL next to WiX EXEs.

Registered with WiX via command line argument to the compiler

Considerations

Before investing in an extension, one should evaluate whether an external tool and the `?include` syntax (from the preprocessor) will provide the needed flexibility for your technical needs. Multiple extensions and extension types are supported but there is no guarantee of the order a particular class of extensions will be processed so there should be no sequencing dependencies between extensions within the same extension class.

Patch Building

Creating a Patch with Wix

Note: You must have Windows Installer 3.0 installed\

1. [Do an administrative install of the RTM version \(Target Image\).](#)
2. [Update Package Id in Main.wxs.](#)
3. [Make new Installer.msi](#)
4. [Do an administrative install of the latest version \(Update Image with new files you want to patch\).](#)
5. [Create a patch creation properties \(.pcp\) file.](#)
6. [Create the patch.](#)
7. [Run the patch](#)

1. Administrative install of RTM version

With the RTM installer.msi, run

```
md c:\patchdir  
md c:\patchdir\rtm  
msiexec /a installer.msi TARGETDIR=c:\patchdir\rtm
```

2. Update Package Id in source .wxs file.

Make a new Guid for the Id= attribute under the `<Package>` tag in the wxs files.

3. Make new installer.msi

4. Administrative install of latest version

With the new installer.msi, run

```
md c:\patchdir\latest  
msiexec /a installer.msi TARGETDIR=c:\patchdir\latest
```

5. Create a Patch Creation Properties (.pcp) file

```
<?xml version="1.0" encoding="utf-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2003/01/wi">
  <?define WixDir = . ?>
  <?define Proj1 = "rtm" ?>
  <?define Proj2 = "latest" ?>

  <PatchCreation
    Id="put-guid-here"
    CleanWorkingFolder="yes"
    OutputPath="patch.pcp"
    WholeFilesOnly="yes"
  >

  <PatchInformation
    Description="Patches the andmagichappens.cmd file"
    Comments="Patch for new dll"
    ShortNames="no"
    Languages="1033"
    Compressed="yes"
    Manufacturer="insert-organization-name-here"/>

  <PatchMetadata
    AllowRemoval="yes"
    Description="Patches the andmagichappens.cmd file"
    ManufacturerName="insert-organization-name-here"
    TargetProductName="insert-product-name-here"
    MoreInfoURL="insert-info-url-here"
    Classification="Hotfix"
    DisplayName="insert-product-abbreviaiton-here Pat

  <Family DiskId="2" MediaSrcProp="insert-product-abbrevia.
    Name="insert-organization-name-here and insert-p
    <UpgradeImage src="$(var.Proj2)\Installer.msi" Id="i
      <TargetImage src="$(var.Proj1)\Installer.msi"
        Id="insert-product-abbreviaito
    </UpgradeImage>
  </Family>

  <PatchSequence
    PatchFamily="insert-organization-name-here and ins
    Target="insert-product-abbreviaiton-hereTarget
```

```
</PatchCreation>  
</Wix>
```

Notes:

The **01** in red in the above file is a patch id. It should be incremented by **1** with each patch. Also, the **Id=** under `<PatchCreation>` should be set to a new Guid for each patch created.

The **SequenceStart** value is influenced by the number of files that the previous patch delivered, as well as the number of files that this patch will deliver. This tells PatchWiz.dll to start assigning File sequence numbers from this number. So if this patch ships 11 files, and the next patch uses a SequenceStart of 1020, it will step on the 11th file's assigned sequence number. In this case the next patch would use a SequenceStart of 1030, and 03 as the patch id to avoid conflicts with this patch. This scheme helps prevent this by coordinating the SequenceStart (file sequence numbers) with the patch sequence number. Also, note that the SequenceStart of the first patch must be greater than the number of files in the original installation. If the original installation contained more than 1000 files(rare), then the SequenceStart for the first patch must be set to a higher value (e.g 2010.)

6. Create the patch msp file

```

candle patch.wxs
light patch.wixobj -out patch.pcp
msimsp -s patch.pcp -p patch.msp -l msimsp.log

```

7.Run the patch

```
msiexec /update patch.msp REINSTALL=ALL /L*v patch.log
```

Using the WixUI dialog library

The WixUI dialog library contains a set of "stock" dialogs providing the familiar wizard-style setup user interface. Several stock dialog sets are supported -- use one **UIRef** to add a user interface to your setup. WixUI is also customizable, from the bitmaps shown in the UI to adding and removing custom dialogs.

Note: The WixUI dialog library is currently at technical preview status. Please provide feedback on the [WiX-devs mailing list](#). Are the provided stock dialog sets useful? Do you have suggestions for others? Hate the UI? Need another dialog? Based on feedback, the WixUI library might change in incompatible ways.

Using the stock dialog sets

The WixUI stock dialog sets support several common dialog sequences:

WixUI_Mondo includes the full set of dialogs (hence "Mondo"): welcome, license agreement, setup type (typical, custom, and complete), feature customization, directory browse, and disk cost. Maintenance-mode dialogs are also included. Use WixUI_Mondo when you have some of your product's features aren't installed by default and there's a meaningful difference between typical and complete installs.

Note: WixUI_Mondo uses [SetInstallLevel](#) control events to set the install level when the user chooses Typical or Complete. For Typical, the install level is set to 3; for Complete, 1000. For details about feature levels and install levels, see [INSTALLLEVEL Property](#).

WixUI_FeatureTree is a simpler version of WixUI_Mondo that omits the setup type dialog. Instead, the user goes directly from the license agreement dialog to the feature customization dialog. WixUI_FeatureTree is more appropriate than WixUI_Mondo when your product installs all features by default.

WixUI_InstallDir doesn't allow the user to choose features but adds a dialog to let the user choose a directory where the product will be installed.

Note: To use WixUI_InstallDir, you must set a property named WIXUI_INSTALLDIR with a value of the ID of the directory you want the user to be able to specify the location of. For example:

```
<Directory Id="TARGETDIR" Name="SourceDir">
  <Directory Id="ProgramFilesFolder" Name="PFiles">
    <Directory Id="TESTFILEPRODUCTDIR" ShortName="WIXTEST" Name
      ...
    </Directory>
  </Directory>
</Directory>
...
<Property Id="WIXUI_INSTALLDIR" Value="TESTFILEPRODUCTDIR" />
<UIRef Id="WixUI_InstallDir" />
```

WixUI_Minimal is the most spartan of the WixUI stock dialog sets. Its sole dialog combines the welcome and license-agreement dialogs and omits the feature customization dialog. WixUI_Minimal is appropriate when your product has no optional features.

How to add a WixUI stock dialog set to a product installer

Assuming you have an existing installer that's functional but just lacking a user interface, here are the steps you need to follow to use a WixUI stock dialog set:

1. Add a UIRef element to your installer source code, using an Id attribute of one of the above dialog sets. For example:

```
<Product ...>  
  <UIRef Id="WixUI_InstallDir" />  
</Product>
```

2. Add wixui.wixlib and the appropriate WixUI localization file to your **light** command line. For example:

```
light Mondo.wixobj %WIXUI_PATH%\WixUI.wixlib -loc %WIXUI_PA
```

For examples, see the .wxs files in the doc/examples/wixui directory.

Specifying a license file

The stock dialog sets have a dialog that displays an end-user license agreement (EULA). To specify your product's license, include a License.rtf file in the current directory when you run **light**. If there isn't such a file, **light** uses the License.rtf file in the ui directory.

Using translated error and progress text

By default, WixUI doesn't include any translated Error or ProgressText elements by default. You can include them by referencing the WixUI_ErrorProgressText UI element:

```
<UIRef Id="WixUI_Minimal" />  
<UIRef Id="WixUI_ErrorProgressText" />
```

Customizing dialog sets

You can most easily add and remove dialogs from the stock dialog sets by copying one of the existing sets and modifying it. For an example, see the project in the `doc/examples/wixui/custom` directory. The following table describes the files:

File name	Description
CustomDialogSet.build	NAnt build file to build the custom dialog set. Builds the WixUI common dialog elements if needed, then builds CustomDialogSet.wxs and CustomDlg.wxs to create CustomDialogSet.wixlib.
CustomDialogSet.wxs	Custom dialog set definition. Copied from WixUI_FeatureTree set and modified to add CustomDlg after the initial WelcomeDlg.
CustomDlg.wxs	Simple custom dialog.
TestCustom.wxs	WiX source code that consumes CustomDialogSet.wixlib.

Replacing the stock bitmaps

The WixUI dialog library includes stock bitmaps for the background of the welcome and installation-complete dialogs and the top banner of the other dialogs. You can "override" those graphics with your own for product-branding purposes. To replace stock bitmaps, add the files from the table below to a subdirectory named Bitmaps under your WiX source file.

File name	Description	Dimensions
bannrbmp.bmp	Top banner	500 × 63
dlgbmp.bmp	Background bitmap used on welcome and install-complete dialogs	503 × 314
exclamic.ico	Exclamation icon on the wait-for-costing dialog	32 × 32
info.ico	Information icon on the cancel and error dialogs	32 × 32
New.ico	Button glyph on directory-browse dialog	16 × 16
Up.ico	Button glyph on directory-browse dialog	16 × 16

Wix Schema

Copyright (c) Microsoft Corporation. All rights reserved. The use and distribution terms for this software are covered by the Common Public License 1.0 (<http://opensource.org/licenses/cpl.php>) which can be found in the file CPL.TXT at the root of this distribution. By using this software in any fashion, you are agreeing to be bound by the terms of this license. You must not remove this notice, or any other, from this software.

Schema for describing Windows Installer database files (.msi/.msm/.pcp).

Root Elements

- [Include](#)
- [Wix](#)

Target Namespace

<http://schemas.microsoft.com/wix/2003/01/wi>

Document Should Look Like

- `<?xml version="1.0"?>`
`<Include xmlns="http://schemas.microsoft.com/wix/2003/01/wi">`
.
.
.
`</Include>`
- `<?xml version="1.0"?>`
`<Wix xmlns="http://schemas.microsoft.com/wix/2003/01/wi">`
.
.
.
`</Wix>`

All Elements

- [AdminExecuteSequence](#)
- [AdminUISequence](#)
- [AdvertiseExecuteSequence](#)

- [AllocateRegistrySpace](#)
- [AppData](#)
- [AppId](#)
- [AppSearch](#)
- [AssemblyName](#)
- [Billboard](#)
- [BillboardAction](#)
- [Binary](#)
- [BindImage](#)
- [Category](#)
- [CCPSearch](#)
- [Certificate](#)
- [CertificateRef](#)
- [Class](#)
- [Column](#)
- [ComboBox](#)
- [ComplianceCheck](#)
- [ComplianceDrive](#)
- [Component](#)
- [ComponentGroup](#)
- [ComponentGroupRef](#)
- [ComponentRef](#)
- [ComponentSearch](#)
- [Condition](#)
- [Configuration](#)
- [ConfigurationData](#)
- [Control](#)
- [CopyFile](#)
- [CostFinalize](#)
- [CostInitialize](#)
- [CreateFolder](#)
- [CreateFolders](#)
- [CreateShortcuts](#)

- [Custom](#)
- [CustomAction](#)
- [CustomActionRef](#)
- [CustomProperty](#)
- [CustomTable](#)
- [Data](#)
- [DeleteServices](#)
- [Dependency](#)
- [Dialog](#)
- [DialogRef](#)
- [DigitalCertificate](#)
- [DigitalSignature](#)
- [Directory](#)
- [DirectoryRef](#)
- [DirectorySearch](#)
- [DirectorySearchRef](#)
- [DisableRollback](#)
- [DuplicateFiles](#)
- [EnsureTable](#)
- [Environment](#)
- [Error](#)
- [Exclusion](#)
- [ExecuteAction](#)
- [Extension](#)
- [ExternalFile](#)
- [Family](#)
- [Feature](#)
- [FeatureRef](#)
- [File](#)
- [FileCost](#)
- [FileSearch](#)
- [FileSearchRef](#)
- [FileShare](#)

- [FileTypeMask](#)
- [FindRelatedProducts](#)
- [ForceReboot](#)
- [Fragment](#)
- [FragmentRef](#)
- [Group](#)
- [GroupRef](#)
- [HTTPHeader](#)
- [Icon](#)
- [IgnoreModularization](#)
- [IgnoreRange](#)
- [Include](#)
- [IniFile](#)
- [IniFileSearch](#)
- [InstallAdminPackage](#)
- [InstallExecute](#)
- [InstallExecuteAgain](#)
- [InstallExecuteSequence](#)
- [InstallFiles](#)
- [InstallFinalize](#)
- [InstallInitialize](#)
- [InstallODBC](#)
- [InstallServices](#)
- [InstallUISequence](#)
- [InstallValidate](#)
- [Interface](#)
- [IsolateComponent](#)
- [IsolateComponents](#)
- [LaunchConditions](#)
- [ListBox](#)
- [ListItem](#)
- [ListView](#)
- [Media](#)

- [Merge](#)
- [MergeRef](#)
- [MigrateFeatureStates](#)
- [MIME](#)
- [MimeMap](#)
- [Module](#)
- [MoveFiles](#)
- [MsiPublishAssemblies](#)
- [MsiUnpublishAssemblies](#)
- [ODBCDataSource](#)
- [ODBCDriver](#)
- [ODBCTranslator](#)
- [Package](#)
- [Patch](#)
- [PatchCertificates](#)
- [PatchCreation](#)
- [PatchFiles](#)
- [PatchInformation](#)
- [PatchMetadata](#)
- [PatchPackage](#)
- [PatchProperty](#)
- [PatchSequence](#)
- [PerfCounter](#)
- [Permission](#)
- [ProcessComponents](#)
- [Product](#)
- [ProgId](#)
- [ProgressText](#)
- [Property](#)
- [PropertyRef](#)
- [ProtectFile](#)
- [ProtectRange](#)
- [Publish](#)

- [PublishComponents](#)
- [PublishFeatures](#)
- [PublishProduct](#)
- [RadioButton](#)
- [RadioButtonGroup](#)
- [RecycleTime](#)
- [RegisterClassInfo](#)
- [RegisterComPlus](#)
- [RegisterExtensionInfo](#)
- [RegisterFonts](#)
- [RegisterMIMEInfo](#)
- [RegisterProduct](#)
- [RegisterProgIdInfo](#)
- [RegisterTypeLibraries](#)
- [RegisterUser](#)
- [Registry](#)
- [RegistrySearch](#)
- [RegistrySearchRef](#)
- [RegistryValue](#)
- [RemoveDuplicateFiles](#)
- [RemoveEnvironmentStrings](#)
- [RemoveExistingProducts](#)
- [RemoveFile](#)
- [RemoveFiles](#)
- [RemoveFolder](#)
- [RemoveFolders](#)
- [RemoveIniValues](#)
- [RemoveODBC](#)
- [RemoveRegistryValues](#)
- [RemoveShortcuts](#)
- [ReplacePatch](#)
- [ReserveCost](#)
- [ResolveSource](#)

- [RMCCPSearch](#)
- [Row](#)
- [ScheduleReboot](#)
- [SelfRegModules](#)
- [SelfUnregModules](#)
- [ServiceArgument](#)
- [ServiceConfig](#)
- [ServiceControl](#)
- [ServiceDependency](#)
- [ServiceInstall](#)
- [SetODBCFolders](#)
- [SFPCatalog](#)
- [SFPFile](#)
- [Shortcut](#)
- [Show](#)
- [SqlDatabase](#)
- [SqlFileSpec](#)
- [SqlLogFileSpec](#)
- [SqlScript](#)
- [SqlString](#)
- [StartServices](#)
- [StopServices](#)
- [Subscribe](#)
- [Substitution](#)
- [SymbolPath](#)
- [TargetFile](#)
- [TargetImage](#)
- [TargetProductCode](#)
- [Text](#)
- [TextStyle](#)
- [TypeLib](#)
- [UI](#)
- [UIRef](#)

- [UIText](#)
- [UnpublishComponents](#)
- [UnpublishFeatures](#)
- [UnregisterClassInfo](#)
- [UnregisterComPlus](#)
- [UnregisterExtensionInfo](#)
- [UnregisterFonts](#)
- [UnregisterMIMEInfo](#)
- [UnregisterProgIdInfo](#)
- [UnregisterTypeLibraries](#)
- [Upgrade](#)
- [UpgradeFile](#)
- [UpgradeImage](#)
- [UpgradeVersion](#)
- [User](#)
- [ValidateProductID](#)
- [Verb](#)
- [WebAddress](#)
- [WebApplication](#)
- [WebApplicationExtension](#)
- [WebAppPool](#)
- [WebDir](#)
- [WebDirProperties](#)
- [WebError](#)
- [WebFilter](#)
- [WebLog](#)
- [WebProperty](#)
- [WebServiceExtension](#)
- [WebSite](#)
- [WebVirtualDir](#)
- [Wix](#)
- [WriteEnvironmentStrings](#)
- [WriteIniValues](#)

- [WriteRegistryValues](#)
- [XmlFile](#)

AdminExecuteSequence Element

Description

None

Windows Installer references

[AdminExecuteSequence Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text (xs:string)

This element may have inner text.

Children

Choice of elements (min: 0, max: unbounded)

- [CostFinalize](#) (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- [CostInitialize](#) (min: 0, max: unbounded): Initiates the internal installation costing process.
- [Custom](#) (min: 0, max: unbounded): Use to sequence a custom action.
- [FileCost](#) (min: 0, max: unbounded): Initiates dynamic costing of standard installation actions.
- [InstallAdminPackage](#) (min: 0, max: unbounded): Copies the product database to the administrative installation point.
- [InstallFiles](#) (min: 0, max: unbounded): Copies files specified in the File table from the source directory to the destination directory.
- [InstallFinalize](#) (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- [InstallInitialize](#) (min: 0, max: unbounded): Marks the beginning of a sequence of actions that change the system.
- [InstallValidate](#) (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.
- [LaunchConditions](#) (min: 0, max: unbounded): Queries the

LaunchCondition table and evaluates each conditional statement recorded there.

- [ResolveSource](#) (min: 0, max: unbounded): Determines the location of the source and sets the SourceDir property if the source has not been resolved yet.

Attributes

None

See Also

[Wix Schema](#)

Version 2.0.4820.0

AdminUISequence Element

Description

None

Windows Installer references

[AdminUISequence Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#), [UI](#)

Inner Text (xs:string)

This element may have inner text.

Children

Choice of elements (min: 0, max: unbounded)

- [CostFinalize](#) (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- [CostInitialize](#) (min: 0, max: unbounded): Initiates the internal installation costing process.
- [Custom](#) (min: 0, max: unbounded): Use to sequence a custom action.
- [ExecuteAction](#) (min: 0, max: unbounded): Initiates the execution sequence.
- [FileCost](#) (min: 0, max: unbounded): Initiates dynamic costing of standard installation actions.
- [InstallAdminPackage](#) (min: 0, max: unbounded): Copies the product database to the administrative installation point.
- [InstallFiles](#) (min: 0, max: unbounded): Copies files specified in the File table from the source directory to the destination directory.
- [InstallFinalize](#) (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- [InstallInitialize](#) (min: 0, max: unbounded): Marks the beginning of a sequence of actions that change the system.
- [InstallValidate](#) (min: 0, max: unbounded): Verifies that all costed

volumes have enough space for the installation.

- [LaunchConditions](#) (min: 0, max: unbounded): Queries the LaunchCondition table and evaluates each conditional statement recorded there.
- [Show](#) (min: 0, max: unbounded)

Attributes

None

See Also

[Wix Schema](#)

Version 2.0.4820.0

AdvertiseExecuteSequence Element

Description

None

Windows Installer references

[AdvtExecuteSequence Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text (xs:string)

This element may have inner text.

Children

Choice of elements (min: 0, max: unbounded)

- [CostFinalize](#) (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- [CostInitialize](#) (min: 0, max: unbounded): Initiates the internal installation costing process.
- [CreateShortcuts](#) (min: 0, max: unbounded): Manages the creation of shortcuts.
- [Custom](#) (min: 0, max: unbounded): Use to sequence a custom action. The only custom actions that are allowed in the AdvtExecuteSequence are type 19 (0x013) type 35 (0x023) and type 51 (0x033).
- [InstallFinalize](#) (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- [InstallInitialize](#) (min: 0, max: unbounded): Marks the beginning of a sequence of actions that change the system.
- [InstallValidate](#) (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.
- [MsiPublishAssemblies](#) (min: 0, max: unbounded): Manages the advertisement of CLR and Win32 assemblies.
- [PublishComponents](#) (min: 0, max: unbounded): Manages the advertisement of the components from the PublishComponent table.

- [PublishFeatures](#) (min: 0, max: unbounded): Writes each feature's state into the system registry.
- [PublishProduct](#) (min: 0, max: unbounded): Manages the advertisement of the product information with the system.
- [RegisterClassInfo](#) (min: 0, max: unbounded): Manages the registration of COM class information with the system.
- [RegisterExtensionInfo](#) (min: 0, max: unbounded): Manages the registration of extension related information with the system.
- [RegisterMIMEInfo](#) (min: 0, max: unbounded): Registers MIME-related registry information with the system.
- [RegisterProgIdInfo](#) (min: 0, max: unbounded): Manages the registration of OLE ProgId information with the system.

Attributes

None

See Also

[Wix Schema](#)

Version 2.0.4820.0

AllocateRegistrySpace Element

Description

Ensures the needed amount of space exists in the registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

AppData Element

Description

Optional way for defining AppData, generally used for complex CDATA.

Windows Installer references

None

Parents

[Category](#)

See Also

[Wix Schema](#)

Version 2.0.4820.0

AppId Element

Description

Application ID containing DCOM information for the associated application GUID. If this element is nested under a Fragment, Module, or Product element, it must be advertised.

Windows Installer references

[AppId Table](#), [Registry Table](#)

Parents

[Component](#), [File](#), [Fragment](#), [Include](#), [Module](#), [Product](#), [TypeLib](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Class](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Uuid	Set this value to the AppID GUID that corresponds to the named executable.	Yes
ActivateAtStorage	YesNoType	Set this value to 'yes' to configure the client to activate on the same system as persistent storage.	
Advertise	YesNoType	Set this value to 'yes' in order to create a normal AppId table row. Set this value to 'no' in order to	

generate Registry rows that perform similar registration (without the often problematic Windows Installer advertising behavior).

Description	String	Set this value to the description of the Appld. It can only be specified when the Appld is not being advertised.
DllSurrogate	String	Set this value to specify that the class is a DLL that is to be activated in a surrogate EXE process, and the surrogate process to be used is the path of a surrogate EXE file specified by the value.
LocalService	String	Set this value to the name of a service to allow the object to be installed as a Win32 service.
RemoteServerName	String	Set this value to the name of the remote server to configure the client to request the object be run at a particular machine whenever an activation function is called for which a COSERVERINFO structure is not specified.

RunAsInteractiveUser [YesNoType](#) Set this value to 'yes' to configure a class to run under the identity of the user currently logged on and connected to the interactive desktop when activated by a remote client without being written as a Win32 service.

ServiceParameters	String	Set this value to the parameters to be passed to a LocalService on invocation.
-------------------	--------	--

Remarks

When being used in unadvertised mode, the attributes in the AppId element correspond to registry keys as follows (values that can be specified in authoring are in bold):

Id

In General

[HKCR\AppID\{**Id**}]

Specific Example

[HKCR\AppID\{**01234567-89AB-CDEF-0123-456789ABCDEF**}]

ActivateAtStorage

In General

[HKCR\AppID\{**Id**}]
ActivateAtStorage="**ActivateAtStorage**"

Specific Example

[HKCR\AppID\{**01234567-89AB-CDEF-0123-456789ABCDEF**}]
ActivateAtStorage="Y"

Description

In General

[HKCR\AppID\{Id}]
@="Description"

Specific Example

[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]
@="My AppId Description"

DllSurrogate

In General

[HKCR\AppID\{Id}]
DllSurrogate="DllSurrogate"

Specific Example

[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]
DllSurrogate="C:\surrogate.exe"

LocalService

In General

[HKCR\AppID\{Id}]
LocalService="LocalService"

Specific Example

[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]
LocalService="MyServiceName"

RemoteServerName

In General

[HKCR\AppID\{Id}]
RemoteServerName="RemoteServerName"

Specific Example

[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]
RemoteServerName="MyRemoteServer"

RunAsInteractiveUser

In General

[HKCR\AppID\{Id}]
RunAs="RunAsInteractiveUser"

Specific Example

```
[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]  
RunAs="Interactive User"
```

ServiceParameters

In General

```
[HKCR\AppID\{Id}]  
ServiceParameters="ServiceParameters"
```

Specific Example

```
[HKCR\AppID\{01234567-89AB-CDEF-0123-456789ABCDEF}]  
ServiceParameters="-param"
```

See Also

[Wix Schema](#)

AppSearch Element

Description

Uses file signatures to search for existing versions of products. The AppSearch action may use this information to determine where upgrades are to be installed. The AppSearch action can also be used to set a property to the existing value of an registry or .ini file entry. AppSearch should be authored into the InstallUISequence table and InstallExecuteSequence table. The installer prevents The AppSearch action from running in the InstallExecuteSequence sequence if the action has already run in InstallUISequence sequence. The AppSearch action searches for file signatures using the CompLocator table first, the RegLocator table next, then the IniLocator table, and finally the DrLocator table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a	

sequence.

Suppress [YesNoType](#) If yes, this action will not occur.

See Also

[Wix Schema](#), [ComponentSearch](#), [FileSearch](#), [IniFileSearch](#),
[RegistrySearch](#)

Version 2.0.4820.0

AssemblyName Element

Description

The MsiAssemblyName table specifies the schema for the elements of a strong assembly cache name for a .NET Framework or Win32 assembly. Consider using the Assembly attribute on File element to have the toolset populate these entries automatically.

Windows Installer references

[MsiAssemblyName Table](#)

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Name of the attribute associated with the value specified in the Value column.	Yes
Value	String	Value associated with the name specified in the Name column.	

See Also

[Wix Schema](#)

Billboard Element

Description

Billboard to display during install of a Feature

Windows Installer references

[Billboard Table](#), [BBControl Table](#)

Parents

[BillboardAction](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Control](#) (min: 0, max: unbounded): Only controls of static type such as: Text, Bitmap, Icon, or custom control can be placed on a billboard.

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for the Billboard.	Yes
Feature	String	Feature whose state determines if the Billboard is shown.	

See Also

[Wix Schema](#)

BillboardAction Element

Description

Billboard action during which child Billboards are displayed

Windows Installer references

[Billboard Table](#), [BBControl Table](#)

Parents

[UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Billboard](#) (min: 1, max: unbounded): Order of Billboard elements determines order of display

Attributes

Name	Type	Description	Required
Id	String	Action name that determines when the Billboard should be shown.	Yes

See Also

[Wix Schema](#)

Binary Element

Description

Binary data used for CustomAction elements and UI controls.

Windows Installer references

[Binary Table](#)

Parents

[Control](#), [Fragment](#), [Include](#), [Module](#), [Product](#), [SqlScript](#), [UI](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The Id cannot be longer than 55 characters. In order to prevent errors in cases where the Id is modularized, it should not be longer than 18 characters.	Yes
SourceFile	String	Path to the binary file.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

See Also

[Wix Schema](#)

BindImage Element

Description

Binds each executable or DLL that must be bound to the DLLs imported by it. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Category Element

Description

Qualified published component for parent Component

Windows Installer references

[PublishComponent Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Sequence (min: 0, max: unbounded)

1. [AppData](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Uuid	A string GUID that represents the category of components being grouped together.	Yes
AppData	String	An optional localizable text describing the category. The string is commonly parsed by the application and can be displayed to the user. It should describe the category.	
Feature	String	Feature that controls the advertisement of the category. Defaults to the primary Feature for the parent Component .	
Qualifier	String	A text string that qualifies the value in the Id attribute. A qualifier is used to distinguish multiple forms of the same Component, such as a Component	Yes

that is implemented in multiple languages.

See Also

[Wix Schema](#)

Version 2.0.4820.0

CCPSearch Element

Description

Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed. The CCPSearch action should be authored into the InstallUISequence table and InstallExecuteSequence table. The installer prevents the CCPSearch action from running in the InstallExecuteSequence sequence if the action has already run in InstallUISequence sequence. The CCPSearch action must come before the RMCCPSearch action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#), [RMCCPSearch](#), [ComplianceCheck](#)

Version 2.0.4820.0

Certificate Element

Description

Used to install and uninstall certificates.

Windows Installer references

None

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for this certificate in the installation package.	Yes
BinaryKey	String	Reference to a Binary element that will store the certificate as a stream inside the package. This attribute cannot be specified with the CertificatePath attribute.	
CertificatePath	String	If the Request attribute is "no" then this attribute is the path to the certificate file outside of the package. If the Request attribute is "yes" then this attribute is the certificate authority to request the certificate from.	

		This attribute may be set via a formatted Property (e.g. [MyProperty]).	
Name	String	Name of the certificate that will be installed or uninstalled in the specified store. This attribute may be set via a formatted Property (e.g. [MyProperty]).	Yes
Overwrite	YesNoType		
PFXPassword	String	If the Binary stream or path to the file outside of the package is a password protected PFX file, the password for that PFX must be specified here. This attribute may be set via a formatted Property (e.g. [MyProperty]).	
Request	YesNoType	This attribute controls whether the CertificatePath attribute is a path to a certificate file (Request='no') or the certificate authority to request the certificate from (Request='yes').	
StoreLocation	Enumeration	This attribute's value should be one of the following: <i>currentUser</i> <i>localMachine</i>	Yes
StoreName	Enumeration	This attribute's value should be one of the following: <i>ca</i> Contains the certificates of certificate authorities that the user trusts to	Yes

issue certificates to others. Certificates in these stores are normally supplied with the operating system or by the user's network administrator.

my

Use the "personal" value instead.

personal

Contains personal certificates. These certificates will usually have an associated private key. This store is often referred to as the "MY" certificate store.

request

root

Contains the certificates of certificate authorities that the user trusts to issue certificates to others. Certificates in these stores are normally supplied with the operating system or by the user's network administrator.

Certificates in this store are typically self-signed.

otherPeople

Contains the certificates of those that the user normally sends

enveloped messages to or receives signed messages from. See [MSDN documentation](#) for more information.

See Also

[Wix Schema](#), [CertificateRef](#)

Version 2.0.4820.0

CertificateRef Element

Description

Associates a certificate with the parent WebSite. The Certificate element should be in the same Component as the parent WebSite.

Windows Installer references

None

Parents

[WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the referenced Certificate.	Yes

See Also

[Wix Schema](#), [Certificate](#)

Class Element

Description

COM Class registration for parent Component.

Windows Installer references

[Class Table](#), [ProgId Table](#), [Registry Table](#), [AppId Table](#)

Parents

[AppId](#), [Component](#), [File](#), [Include](#), [TypeLib](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [FileTypeMask](#) (min: 0, max: unbounded)
- [Interface](#) (min: 0, max: unbounded): These Interfaces will be registered with the parent Class and TypeLib (if present).
- [ProgId](#) (min: 0, max: unbounded): A ProgId associated with Class must be a child element of the Class element

Attributes

Name	Type	Description
Id	Uuid	The Class identifier (CLSID) of a COM server.
Advertise	YesNoType	Set this value to "yes" in order to create a normal Class table row. Set this value to "no" in order to generate Registry rows that perform similar registration (without the problematic Windows Installer advertise behavior).
AppId	Uuid	This attribute is only allowed when a class is advertised. Using this attribute will register an Application ID containing DCOM information for the associated application.

GUID. The value must correspond to `ApplId/@Id` of an `ApplId` element nested within a `Fragment`, `Module`, or `Product` element. To associate an `ApplId` with a non-advertised class, nest the class within a parent `Class` element.

Argument	String	This column is optional only when the <code>Context</code> column is set to "LocalServer" or "LocalServer32" server context. The value is registered as the argument against the server and is used by OLE for invoking the server. Note that the resolution of properties in the <code>Argument</code> field is limited. A property formatted as <code>[Property]</code> in this field can be resolved if the property already has an intended value when the component containing the class is installed. For example, for an argument <code>"[#MyDoc.doc]"</code> to resolve to the correct value, the same process must be used for installing the file <code>MyDoc.doc</code> and the component that owns the class.
Context	List	The server context(s) for this server. The attribute's value should be a space-delimited list containing one or more of the following: <i>LocalServer</i> A 16-bit local server application. <i>LocalServer32</i> A 32-bit local server application. <i>InprocServer</i> A 16-bit in-process server DLL. <i>InprocServer32</i> A 32-bit in-process server DLL.
Control	YesNoType	Set this attribute's value to 'yes' to identify the object as an ActiveX Control. The default value is 'no'. See http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ole/ole_ole32.asp

url=/library/en-us/com/htm/reg_2fn0.a
 more information.

Description	String	Localized description associated with Class ID and Program ID.
Handler	String	The default inproc handler. May be provided only for Context = LocalServer32. Value of "1" creates a 16-bit InprocHandler (appearing as the InprocHandler value). Value of "2" creates a 32-bit InprocHandler (appearing as the InprocHandler32 value). Value of "3" creates a 16-bit as well as 32-bit InprocHandler. A non-numeric value is treated as a system file that serves as the 32-bit InprocHandler (appearing as the InprocHandler32 value).
Icon	String	The file providing the icon associated with this CLSID. Reference to an Icon element (should match the Id attribute of an Icon element). This is currently not supported. The value of the Advertise attribute is ignored.
IconIndex	Integer	Icon index into the icon file.
Insertable	YesNoType	Specifies the CLSID may be insertable.
Programmable	YesNoType	Specifies the CLSID may be programmable.
RelativePath	YesNoType	When the value is "yes", the bare file name can be used for COM servers. The inproc handler registers the file name only instead of a complete path. This enables the server to use the current directory to take precedence over the system directory. This also allows multiple copies of the same component.
SafeForInitializing	YesNoType	May only be specified if the value of the Advertise attribute is "no".
SafeForScripting	YesNoType	May only be specified if the value of the Advertise attribute is "no".
Server	String	May only be specified if the value of the Advertise attribute is "no". File Id of the server.

		server file. If this element is nested under a File element, this value defaults to the value of the parent File/@Id.
ThreadingModel	Enumeration	Threading model for the CLSID. This attribute's value should be one of the following: <i>apartment</i> <i>free</i> <i>both</i> <i>neutral</i> <i>single</i> <i>rental</i>
Version	String	Version for the CLSID.

Remarks

When being used in unadvertised mode, the attributes in the Class element correspond to registry keys as follows (values that can be specified in authoring are in bold):

Id/Context/Server

In General

[HKCR\CLSID\{Id}\Context1]

@="[!Server]"

[HKCR\CLSID\{Id}\Context2]

@="[!Server]"

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\LocalServer]

@="[!comserv.dll]"

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\LocalServer32]

@="[!comserv.dll]"

AppId

In General

[HKCR\CLSID\{Id}]

AppId="{AppId}"

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}]

AppId="{00000000-89AB-0000-0123-000000000000}"

Argument

In General

[HKCR\CLSID\{Id}\Context]

@="[!Server] Argument"

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\LocalServer32]

@="[!comserv.dll] /arg1 /arg2 /arg3"

Control

In General

Value "yes" specified:

[HKCR\CLSID\{Id}\Control]

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\Control]

Description

In General

[HKCR\CLSID\{Id}]

@="Description"

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}]

@="Description of Example COM Component"

Handler

In General

Value "1" specified:

[HKCR\CLSID\{Id}\InprocHandler]

@="ole.dll"

Value "2" specified:

```
[HKCR\CLSID\{Id}\InprocHandler32]
@="ole32.dll"
Value "3" specified:
[HKCR\CLSID\{Id}\InprocHandler]
@="ole.dll"
[HKCR\CLSID\{Id}\InprocHandler32]
@="ole32.dll"
Other value specified:
[HKCR\CLSID\{Id}\InprocHandler32]
@="Handler"
```

Specific Example (for other value)

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}\InprocHandler32]
@="handler.dll"
```

Icon/IconIndex

This is not currently handled properly.

Insertable

In General

```
Value "no" specified:
[HKCR\CLSID\{Id}\NotInsertable]
Value "yes" specified:
[HKCR\CLSID\{Id}\Insertable]
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}\Insertable]
```

Programmable

In General

```
Value "yes" specified:
[HKCR\CLSID\{Id}\Programmable]
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}\Programmable]
```

RelativePath

Unsupported. Please contribute this back to WiX if you know.

SafeForInitializing

In General

Value "yes" specified:

```
[HKCR\CLSID\{Id}\Implemented Categories\{7DD95802-9882-11CF-9FA9-00AA006C42C4}]
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\Implemented Categories\{7DD95802-9882-11CF-9FA9-00AA006C42C4}]
```

SafeForScripting

In General

Value "yes" specified:

```
[HKCR\CLSID\{Id}\Implemented Categories\{7DD95801-9882-11CF-9FA9-00AA006C42C4}]
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\Implemented Categories\{7DD95801-9882-11CF-9FA9-00AA006C42C4}]
```

ThreadingModel

In General

```
[HKCR\CLSID\{Id}\Context]  
ThreadingModel="ThreadingModel"
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\LocalServer32]  
ThreadingModel="Apartment"
```

TypeLibId (from parent TypeLib/@Id)

In General

```
[HKCR\CLSID\{Id}\TypeLib]  
@="{TypeLibId}"
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\TypeLib]  
@="{11111111-89AB-1111-0123-111111111111}"
```

Version

In General

```
[HKCR\CLSID\{Id}\Version]  
@="Version"
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-  
456789ABCDEF}\Version]  
@="1.0.0.0"
```

See Also

[Wix Schema](#), [AppId](#)

Version 2.0.4820.0

Column Element

Description

Column definition for a Custom Table

Windows Installer references

None

Parents

[CustomTable](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the column.	Yes
Category	Enumeration	Category of this column. This attribute's value should be one of the following: <i>Text</i> <i>UpperCase</i> <i>LowerCase</i> <i>Integer</i> <i>DoubleInteger</i> <i>TimeDate</i> <i>Identifier</i> <i>Property</i> <i>Filename</i>	

WildcardFilename

Path

Paths

AnyPath

DefaultDir

RegPath

Formatted

Template

Condition

Guid

Version

Language

Binary

CustomSource

Cabinet

Shortcut

Description	String	Description of this column.
KeyColumn	Integer	Column in the table in KeyTable attribute.
KeyTable	String	Table in which this column is an external key. Can be semicolon delimited.
Localizable	YesNoType	Whether this column can be localized.
MaxValue	Integer	Maximum value for a numeric value, date or version in this column.
MinValue	Integer	Minimum value for a numeric value, date or version in this

column.

Modularize	ModularizeType	How this column should be modularized, if at all.	
Nullable	YesNoType	Whether this column can be left null.	
PrimaryKey	YesNoType	Whether this column is a primary key.	
Set	String	Semicolon delimited list of permissible values.	
Type	Enumeration	The type of this column. This attribute's value should be one of the following: <i>binary</i> <i>int</i> <i>string</i>	Yes
Width	Integer	Width of this column.	

See Also

[Wix Schema](#)

ComboBox Element

Description

Set of items for a particular ComboBox control tied to an install Property

Windows Installer references

[ComboBox Table](#), [Control Table](#), [Dialog Table](#)

Parents

[Control](#), [UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ListItem](#) (min: 0, max: unbounded): entry for ComboBox table

Attributes

Name	Type	Description	Required
Property	String	Property tied to this group	Yes

See Also

[Wix Schema](#)

ComplianceCheck Element

Description

Adds a row to the CCPSearch table.

Windows Installer references

[CCPSearch Table](#), [Signature Table](#)

Parents

[Fragment](#), [Include](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ComplianceDrive](#) (min: 0, max: 1): Starts searches from the CCP_DRIVE.
2. [ComponentSearch](#) (min: 0, max: unbounded)
3. [RegistrySearch](#) (min: 0, max: unbounded)
4. [IniFileSearch](#) (min: 0, max: unbounded)
5. [DirectorySearch](#) (min: 0, max: unbounded)
6. [FileSearch](#) (min: 0, max: unbounded)

Attributes

None

See Also

[Wix Schema](#), [Property](#)

ComplianceDrive Element

Description

Sets the parent of a nested DirectorySearch element to CCP_DRIVE.

Windows Installer references

None

Parents

[ComplianceCheck](#), [Property](#)

Inner Text

None

Children

Choice of elements (min: 1, max: 1)

- [DirectorySearch](#) (min: 1, max: 1)
- [DirectorySearchRef](#) (min: 1, max: 1)

Attributes

None

See Also

[Wix Schema](#)

Component Element

Description

Component for parent Directory

Windows Installer references

[Component Table](#), [Condition Table](#), [Directory Table](#)

Parents

[Directory](#), [DirectoryRef](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [AppId](#) (min: 0, max: unbounded)
- [Category](#) (min: 0, max: unbounded)
- [Certificate](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [Condition](#) (min: 0, max: unbounded)
- [CopyFile](#) (min: 0, max: unbounded)
- [CreateFolder](#) (min: 0, max: unbounded)
- [Environment](#) (min: 0, max: unbounded)
- [Extension](#) (min: 0, max: unbounded)
- [File](#) (min: 0, max: unbounded)
- [FileShare](#) (min: 0, max: unbounded)
- [IniFile](#) (min: 0, max: unbounded)
- [Interface](#) (min: 0, max: unbounded)
- [IsolateComponent](#) (min: 0, max: unbounded)
- [ODBCDataSource](#) (min: 0, max: unbounded)
- [ODBCDriver](#) (min: 0, max: unbounded)
- [ODBCTranslator](#) (min: 0, max: unbounded)
- [ProgId](#) (min: 0, max: unbounded)

- [Registry](#) (min: 0, max: unbounded)
- [RemoveFile](#) (min: 0, max: unbounded)
- [RemoveFolder](#) (min: 0, max: unbounded)
- [ReserveCost](#) (min: 0, max: unbounded)
- [ServiceConfig](#) (min: 0, max: unbounded)
- [ServiceControl](#) (min: 0, max: unbounded)
- [ServiceInstall](#) (min: 0, max: unbounded)
- [Shortcut](#) (min: 0, max: unbounded)
- [SqlDatabase](#) (min: 0, max: unbounded)
- [SqlScript](#) (min: 0, max: unbounded)
- [SqlString](#) (min: 0, max: unbounded)
- [TypeLib](#) (min: 0, max: unbounded)
- [User](#) (min: 0, max: unbounded)
- [WebAppPool](#) (min: 0, max: unbounded)
- [WebDir](#) (min: 0, max: unbounded)
- [WebFilter](#) (min: 0, max: unbounded)
- [WebProperty](#) (min: 0, max: unbounded)
- [WebServiceExtension](#) (min: 0, max: unbounded)
- [WebSite](#) (min: 0, max: unbounded)
- [WebVirtualDir](#) (min: 0, max: unbounded)
- [XmlFile](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description
Id	String	Component identifier; t components.
ComPlusFlags	Integer	Set this attribute to cre should be the export fla the .msi file. For more i documentation in the P
DisableRegistryReflection	YesNoType	Set this attribute to 'yes reflection on all existing this component. When

		calls the RegDisableRe accessed by the comp Windows Installer versi systems.
DiskId	String	This attribute must be s or all of its children File attribute should corres element authored elsev between a component packaging options to th element (values such a embedding, etc...).
DriverAddRemovePrograms	YesNoType	Specifies that the DIFx entry in the Add/Remov The default is 'yes'.
DriverDeleteFiles	YesNoType	If set to "yes", configure that were copied to the a driver package was ir "no" or not present, DIF from a system. Note th these files is controlled component that repres MsiDriverPackages cus DriverDeleteFiles to "ye Flags entry value. Setti corresponding bit in the is not present, DIFxApp
DriverForceInstall	YesNoType	Specifies that the DIFx the installation of a new even if the currently ins better match than the n excellent way to ensure recognize the Compon The default is null whic install a driver via DIFx http://www.microsoft.co for more information.
DriverLegacy	YesNoType	If set to "yes", configure

driver packages and dr
 For more information, s
 Packages in Legacy M
 attribute is set to "no" o
 only signed driver pack
 DIFxApp to install unsiq
 Flags entry value of the
 driver package in the M
 Setting DriverLegacy to
 the Flags entry value. S
 the bit in the Flags entr
 install unsigned driver p
 present, DIFxApp uses

DriverPlugAndPlayPrompt	YesNoType	Specifies that the DIFx, the user to connect the connected. The default
DriverSequence	Integer	Specifies an optional in DIFxApp CustomAction installation package in numbers. The same se more than one driver; h packages with the sam installed cannot be dete
Guid	ComponentGuid	This value should be a component's contents, It's also possible to set specify an unmanaged components are a secu component cannot be r Installer (it is essentially component). Therefore for any component whi need to be patched in t
KeyPath	YesNoType	If this attribute's value i this Component is usec key or File as the KeyP KeyPath attribute to 'ye
Location	Enumeration	This attribute's value sh

local
Prevents the component from being installed on the network (this attribute is not set)

source
Enforces that the component has a source (it cannot be installed from the local cache)

either
Allows the component to be installed either locally or on the network

NeverOverwrite	YesNoType	If this attribute is set to Yes, the component cannot be overwritten or reinstall the component registry entry for the component. If the application does register the component, use this flag to prevent the component from being registered by the component. Use this flag for components registered with ProgId, MIME, and Ver
Permanent	YesNoType	If this attribute is set to Yes, the component is installed during an extra system client for an installer registry setting at least one product is installed (component). Note that if you do not set a guid to be permanent, it is still permanent (it still tracks it), it's just not
SharedDllRefCount	YesNoType	If this attribute's value is Yes, the installer increments the reference count of the component's key. If the installer increments the reference count already
Transitive	YesNoType	If this attribute is set to Yes, the value of the statement is the value was previously True and the installer installs the component previously True and has

Win64	YesNoType	removes the componer products as clients. Set this attribute to 'yes' component. This attribute packages that include l If this bit is not set, the component. If this is a (component, set this bit attribute.
-------	---------------------------	--

[Any attribute namespace='###other' processContents='lax'](#)

See Also

[Wix Schema](#), [ComponentRef](#), [Media](#)

Version 2.0.4820.0

ComponentGroup Element

Description

Groups together multiple components to be used in other locations.

Windows Installer references

None

Parents

[Fragment](#)

Inner Text

None

Children

Choice of elements (min: 1, max: unbounded)

- [ComponentRef](#) (min: 1, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the ComponentGroup.	Yes
Any attribute namespace='##other' processContents='lax'			

See Also

[Wix Schema](#), [ComponentGroupRef](#)

Version 2.0.4820.0

ComponentGroupRef Element

Description

Create a reference to a ComponentGroup in another Fragment.

Windows Installer references

None

Parents

[Feature](#), [FeatureRef](#), [Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the ComponentGroup to reference.	Yes
Primary	YesNoType	Set this attribute to 'yes' in order to make the parent feature of this component the primary feature for this component. Components may belong to multiple features. By designating a feature as the primary feature of a component, you ensure that whenever a component is selected for install-on-demand (IOD), the primary feature will be the one to install it. This attribute should only be set if a component actually nests under multiple features. If a component nests under only one feature, that feature is the primary feature for the	

component. You cannot set more than one feature as the primary feature of a given component.

See Also

[Wix Schema](#), [ComponentGroup](#)

Version 2.0.4820.0

ComponentRef Element

Description

Create a reference to a Feature element in another Fragment.

Windows Installer references

None

Parents

[ComponentGroup](#), [Feature](#), [FeatureRef](#), [Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Component element to reference.	Yes
Primary	YesNoType	Set this attribute to 'yes' in order to make the parent feature of this component the primary feature for this component. Components may belong to multiple features. By designating a feature as the primary feature of a component, you ensure that whenever a component is selected for install-on-demand (IOD), the primary feature will be the one to install it. This attribute should only be set if a component actually nests under multiple features. If a component nests under only one feature, that feature is the primary feature for the	

component. You cannot set more than one feature as the primary feature of a given component.

See Also

[Wix Schema](#), [Component](#)

Version 2.0.4820.0

ComponentSearch Element

Description

Searches for file or directory and assigns to value of parent Property.

Windows Installer references

[CompLocator Table](#), [Signature Table](#)

Parents

[ComplianceCheck](#), [Property](#)

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [DirectorySearch](#) (min: 0, max: 1)
- [DirectorySearchRef](#) (min: 0, max: 1)
- [FileSearch](#) (min: 0, max: 1)
- [FileSearchRef](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String		Yes
Guid	Uuid	The component ID of the component whose key path is to be used for the search.	
Type	Enumeration	Must be file if last child is FileSearch element and must be directory if last child is DirectorySearch element. This attribute's value should be one of the following: <i>directory</i> <i>file</i>	

See Also

[Wix Schema](#), [IniFileSearch](#), [RegistrySearch](#)

Version 2.0.4820.0

Condition Element

Description

Conditions for components, controls, features, and products. The condition is specified in the inner text of the element.

Windows Installer references

[Component Table](#), [ControlCondition Table](#), [Condition Table](#), [LaunchCondition Table](#)

Parents

[Component](#), [Control](#), [Feature](#), [Fragment](#), [Include](#), [Product](#)

Inner Text (xs:string)

Under a Component element, the condition becomes the condition of the component. Under a Control element, the condition becomes a ControlCondition entry. Under a Feature element, the condition becomes a Condition entry. Under a Fragment or Product element, the condition becomes a LaunchCondition entry.

Children

None

Attributes

Name	Type	Description	Required
Action	Enumeration	Used only under Control elements and is required. Allows specific actions to be applied to a control based on the result of this condition. This attribute's value should be one of the following: <i>default</i> <i>enable</i> <i>disable</i> <i>hide</i>	

show

Level	String	Used only under Feature elements and is required. Allows modifying the level of a Feature based on the result of this condition.
Message	String	Used only under Fragment or Product elements and is required. Set the value to the text to display when the condition fails and the installation must be terminated.

See Also

[Wix Schema](#)

Configuration Element

Description

Defines the configurable attributes of merge module.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
ContextData	String	Specifies a semantic context for the requested data.	
DefaultValue	String	Specifies a default value for the item in this record if the merge tool declines to provide a value.	
Description	String	Description for authoring.	
DisplayName	String	Display name for authoring.	
Format	Enumeration	Specifies the format of the data being changed. This attribute's value should be one of the following: <i>Text</i> <i>Key</i> <i>Integer</i>	Yes

Bitfield

HelpKeyword	String	Keyword into chm file for authoring.	
HelpLocation	String	Location of chm file for authoring.	
KeyNoOrphan	YesNoType	Does not merge rule according to rules in MSI SDK.	
Name	String	Defines the name of the configurable item.	Yes
NonNullable	YesNoType	If yes, null is not a valid entry.	
Type	String	Specifies the type of the data being changed.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

ConfigurationData Element

Description

Data to use as input to a configurable merge module.

Windows Installer references

None

Parents

[Merge](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Name	String	Key into the ModuleConfiguration table.	Yes
Value	String	Value to be passed to configurable merge module.	Yes

See Also

[Wix Schema](#)

Control Element

Description

None

Windows Installer references

[Control Table](#), [ComboBox Table](#), [Dialog Table](#), [ListBox Table](#), [ListView Table](#), [RadioButton Table](#)

Parents

[Billboard](#), [Dialog](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Text](#) (min: 0, max: 1): alternative to Text attribute when CDATA is needed to escape XML delimiters
2. [ComboBox](#) (min: 0, max: 1): ComboBox table with ListItem children
3. [ListBox](#) (min: 0, max: 1): ListBox table with ListItem children
4. [ListView](#) (min: 0, max: 1): ListView table with ListItem children
5. [RadioButtonGroup](#) (min: 0, max: 1): RadioButton table with RadioButton children
6. [Property](#) (min: 0, max: 1): Property table entry for the Property table column associated with this control
7. [Binary](#) (min: 0, max: 1): Icon referenced in icon column of row
8. Choice of elements (min: 0, max: unbounded)
 - [Condition](#) (min: 0, max: unbounded): Condition to specify actions for this control based on the outcome of the condition.
 - [Publish](#) (min: 0, max: unbounded)

- [Subscribe](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
Bitmap	YesNoType	only valid for RadioButton and PushButton Controls	
Cancel	YesNoType	yes if this is the control invoked on cancel of dialog	
CDROM	YesNoType	only valid for Volume and Directory Controls	
CheckBoxValue	String	Only for CheckBox control to set Property to a value on check	
ComboList	YesNoType	only valid for ComboBox Controls	
Default	YesNoType	yes if this control invoked by the return key	
Disabled	YesNoType		
ElevationShield	YesNoType	Only valid for PushButton controls. This attribute adds the User Account Control (UAC) elevation icon (shield icon) to the PushButton control. If this attribute's value is	

'yes' and the installation is not yet running with elevated privileges, the pushbutton control is created using the User Account Control (UAC) elevation icon (shield icon). If this attribute's value is 'yes' and the installation is already running with elevated privileges, the pushbutton control is created using the other icon attributes. Otherwise, the pushbutton control is created using the other icon attributes.

Fixed	YesNoType	only valid for Volume and Directory Controls
FixedSize	YesNoType	only valid for RadioButton, PushButton, and Icon Controls
Floppy	YesNoType	only valid for Volume and Directory Controls
FormatSize	YesNoType	only valid for Text Controls

HasBorder	YesNoType	only valid for RadioButton Controls	
Height	LocalizableInteger		Yes
Help	String		
Hidden	YesNoType		
Icon	YesNoType	only valid for RadioButton and PushButton Controls	
IconSize	Enumeration	only valid for RadioButton, PushButton, and Icon Controls This attribute's value should be one of the following: 16 32 48	
Image	YesNoType	only valid for RadioButton, PushButton, and Icon Controls	
Indirect	YesNoType		
Integer	YesNoType		
LeftScroll	YesNoType		
Multiline	YesNoType	only valid for Edit Controls	
NoPrefix	YesNoType	only valid for Text Controls	
NoWrap	YesNoType	only valid for Text Controls	
Password	YesNoType	only valid for Edit	

		Controls
ProgressBlocks	YesNoType	only valid for ProgressBar Controls
Property	String	
PushLike	YesNoType	only valid for RadioButton and Checkbox Controls
RAMDisk	YesNoType	only valid for Volume and Directory Controls
Remote	YesNoType	only valid for Volume and Directory Controls
Removable	YesNoType	only valid for Volume and Directory Controls
RightAligned	YesNoType	
RightToLeft	YesNoType	
ShowRollbackCost	YesNoType	only valid for VolumeCostList Controls
Sorted	YesNoType	This attribute is only valid for Listbox, ListView, and ComboBox Controls. Set the value of this attribute to "yes" to have entries appear in the order specified under the Control. If the attribute value is "no" or absent the entries in the

		control will appear in alphabetical order.	
Sunken	YesNoType		
TabSkip	YesNoType	yes if this control is skipped in the tab sequence	
Text	String		
ToolTip	String		
Transparent	YesNoType	only valid for Text Controls	
Type	String		Yes
UserLanguage	YesNoType	only valid for Text Controls	
Width	LocalizableInteger		Yes
X	LocalizableInteger		Yes
Y	LocalizableInteger		Yes

See Also

[Wix Schema](#)

CopyFile Element

Description

Copy or move an existing file on the target machine, or copy a file that is being installed, to another destination. When this element is nested under a File element, the parent file will be installed, then copied to the specified destination if the parent component of the file is selected for installation or removal. When this element is nested under a Component element and no FileId attribute is specified, the file to copy or move must already be on the target machine. When this element is nested under a Component element and the FileId attribute is specified, the specified file is installed, then copied to the specified destination if the parent component is selected for installation or removal (use this option to control the copy of a file in a different component by the parent component's installation state). If the specified destination directory is the same as the directory containing the original file and the name for the proposed source file is the same as the original, then no action takes place.

Windows Installer references

[DuplicateFile Table](#), [MoveFile Table](#)

Parents

[Component](#), [File](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
Id	String	Primary key use identify this particular entry.
Delete	YesNoType	This attribute ca

be specified if the element is nested under a File element or the FileId attribute is specified. In other cases, if the attribute is not specified, the default value is "no" and the file is copied, not moved. Set the value to "yes" in order to move the file (thus deleting the source file) instead of copying.

DestinationDirectory	String	Set this value to the destination directory where an existing file on the target machine should be moved or copied. This directory must exist in the installation database at creation time. This attribute cannot be specified in conjunction with DestinationProperty.
DestinationLongName	LongFileNameType	If the destination name of the file needs to be longer than 8.3 format, then this attribute should be specified with the long file name (in addition to the Name attribute).

		which is always required for target systems that might not support long names).
DestinationName	ShortFileNameType	Set this value to localizable name be given to the original file after moved or copied this attribute is not specified, then the destination file is given the same name as the source file.
DestinationProperty	String	Set this value to property that will have a value that resolves to the full path of the destination directory. The property does not have to exist in the installer database at creation time; it could be created at installation time or by a custom action, or the command line etc. This attribute cannot be specified in conjunction with DestinationDirectory.
FileId	String	This attribute can be specified if the element is nested

SourceDirectory	String	<p>under a File element. Set this attribute's value to the identifier of a component from a different component to copy it based on the install state of the parent component.</p> <p>This attribute can be specified if the element is nested under a File element or the FileId attribute is specified. Set the value to the source directory from which to copy or move an existing file on the target machine. The directory must exist in the installer database at creation time. This attribute cannot be specified in conjunction with SourceProperty.</p>
SourceName	WildcardLongFileNameType	<p>This attribute can be specified if the element is nested under a File element or the FileId attribute is specified. Set the value to the localizable name of the file(s) to be copied or moved of the files that</p>

match the wild c
 will be removed
 from the specifie
 directory. The va
 is a filename tha
 may also contain
 wild card charac
 "?" for any single
 character or "*" f
 zero or more
 occurrences of a
 character. If this
 attribute is not
 specified (and th
 element is not
 nested under a F
 element or speci
 Filed attribute) t
 the SourcePrope
 attribute should l
 set to the name o
 property that will
 resolve to the ful
 path of the sourc
 filename. If the
 value of this
 attribute contains
 "*" wildcard and
 DestinationName
 attribute is speci
 all moved or cop
 files retain the fil
 names from their
 sources.

SourceProperty

String

This attribute can
 be specified if th
 element is neste
 under a File eler

or the FileId attribute is specified. Set the value to a property that will have a value that resolves to the full path of the source directory, full path including file name if SourceName is not specified). The property does not have to exist in the installer database at creation time; it could be created at installation time by a custom action, or the command line, etc. This attribute cannot be specified in conjunction with SourceDirectory.

See Also

[Wix Schema](#), [RemoveFile](#)

Version 2.0.4820.0

CostFinalize Element

Description

Ends the internal installation costing process begun by the CostInitialize action. Any standard or custom actions that affect costing should be sequenced before the CostInitialize action. Call the FileCost action immediately following the CostInitialize action and then call the CostFinalize action to make all final cost calculations available to the installer through the Component table. The CostFinalize action must be executed before starting any user interface sequence which allows the user to view or modify Feature table selections or directories. The CostFinalize action queries the Condition table to determine which features are scheduled to be installed. Costing is done for each component in the Component table. The CostFinalize action also verifies that all the target directories are writable before allowing the installation to continue. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#),
[InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a	

sequence.

Suppress [YesNoType](#) If yes, this action will not occur.

See Also

[Wix Schema](#), [CostInitialize](#), [FileCost](#)

Version 2.0.4820.0

CostInitialize Element

Description

Initiates the internal installation costing process. Any standard or custom actions that affect costing should be sequenced before the CostInitialize action. Call the FileCost action immediately following the CostInitialize action. Then call the CostFinalize action following the CostInitialize action to make all final cost calculations available to the installer through the Component table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#),
[InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#), [FileCost](#), [CostFinalize](#)

CreateFolder Element

Description

Create folder as part of parent Component.

Windows Installer references

[CreateFolder Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Permission](#) (min: 0, max: unbounded): ACL permission
- [Shortcut](#) (min: 0, max: unbounded): Non-advertised shortcut to this folder, Shortcut Target is preset to the folder

Attributes

Name	Type	Description	Required
Directory	String	Identifier of Directory to create. Defaults to Directory of parent Component.	

See Also

[Wix Schema](#), [RemoveFolder](#)

CreateFolders Element

Description

Creates empty folders for components that are set to be installed. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

CreateShortcuts Element

Description

Manages the creation of shortcuts. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Custom Element

Description

Use to sequence a custom action.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#),
[InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
Action	String		Yes
After	String		
Before	String		
OnExit	Enumeration	mutually exclusive with Before, After, and Sequence attributes This attribute's value should be one of the following: <i>success</i> <i>cancel</i> <i>error</i> <i>suspend</i>	
Sequence	Integer		

See Also

[Wix Schema](#), [CustomAction](#)

Version 2.0.4820.0

CustomAction Element

Description

Specifies a custom action to be added to the MSI CustomAction table. Various combinations of the attributes for this element correspond to different custom action types. For more information about custom actions see the MSDN documentation http://msdn.microsoft.com/library/en-us/msi/setup/summary_list_of_all_custom_action_types.asp for a "Summary List of All Custom Action Types".

Windows Installer references

[CustomAction Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text (xs:string)

The text node is only valid if the Script attribute is specified. In that case, the text node contains the script to embed.

Children

None

Attributes

Name	Type	Description
BinaryKey	String	This attribute is a reference to a E stream contains the custom action installed into a target directory. Th specify the custom action DLL to attribute to specify a type 17 cust the VBScriptCall or JScriptCall at
Directory	String	This attribute specifies a referenc containing a directory path. This a attribute to specify the source exe attribute to specify a formatted str type 35 custom action.

DllEntry	String	This attribute specifies the name attribute is used with the BinaryKey FileKey attribute to create a type
Error	String	This attribute specifies an index in type 19 custom action that display
ExeCommand	String	This attribute specifies the comm: executable. This attribute is typical custom action, the FileKey attribute a type 50 custom action, or the D the executable to run.
Execute	Enumeration	<p>This attribute indicates the schedule one of the following:</p> <p><i>commit</i> Indicates that the custom act installation script (at the end</p> <p><i>deferred</i> Indicates that the custom act</p> <p><i>firstSequence</i> Indicates that the custom act</p> <p><i>immediate</i> Indicates that the custom act privileges. This is the default.</p> <p><i>oncePerProcess</i> Indicates that the custom act same process.</p> <p><i>rollback</i> Indicates that a custom actio during installation, usually to</p> <p><i>secondSequence</i> Indicates that a custom actio an earlier sequence.</p>
FileKey	String	This attribute specifies a reference execute the custom action code it typically used with the ExeComm

		runs an installed executable, with action DLL to use for a type 17 custom action. The value of this attribute specifies a type 21 or 22 custom action.
HideTarget	YesNoType	Ensures the installer does not log
Id	String	The identifier of the custom action
Impersonate	YesNoType	This attribute specifies whether the custom action should impersonate the user context of the custom action. Typically the value should be Yes to request elevated privileges to apply changes.
JScriptCall	String	This attribute specifies the name of the JavaScript file that must be provided in a Binary element above. In other words, this attribute specifies the file name.
Property	String	This attribute specifies a reference to a property that specifies the Property to be used in the custom action. This attribute is typically used with the Value attribute. The text in Value and placed in the custom action is used with the ExeCommand attribute. The value of the given property to specify the value of the custom action are often useful to pass values to the custom action. For more information, see http://msdn.microsoft.com/library/us/msi/setup/obtaining_context_in_installer.asp .
Return	Enumeration	Set this attribute to set the return code of the custom action should be one of the following: <i>asyncNoWait</i> Indicates that the custom action should not wait for the installer to terminate. <i>asyncWait</i> Indicates that the custom action should wait for the return code at sequence completion. <i>check</i> Indicates that the custom action should be checked for success. This is the default value. <i>ignore</i> Indicates that the custom action should not be checked for success.

be checked.

Script	Enumeration	Creates a type 37 or 38 custom action to be embedded in the package. <i>jscript</i> <i>vbscript</i>
TerminalServerAware	YesNoType	This attribute specifies controls w user during per-machine installs c custom actions that do not specify user impersonation on Terminal S attribute is only applicable when i
Value	String	This attribute specifies a string va used with the Property attribute to with the Directory attribute to set ; The value can be a literal value o syntax.
VBScriptCall	String	This attribute specifies the name ; script must be provided in a Binary described above. In other words, BinaryKey attribute.
Win64	YesNoType	Specifies that a script custom acti the Script, VBScriptCall, and JScr

See Also

[Wix Schema](#), [Custom](#), [CustomActionRef](#)

CustomActionRef Element

Description

This will cause the entire contents of the Fragment containing the referenced CustomAction to be included in the installer database.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the CustomAction to reference.	Yes

Any attribute namespace='###other' processContents='lax'

See Also

[Wix Schema](#), [CustomAction](#)

CustomProperty Element

Description

A custom property for the PatchMetadata table.

Windows Installer references

None

Parents

[PatchMetadata](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Company	String	The name of the company.	Yes
Property	String	The name of the metadata property.	Yes
Value	String	Value of the metadata property.	Yes

See Also

[Wix Schema](#)

CustomTable Element

Description

Defines a custom table for use from a custom action.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Column](#) (min: 0, max: unbounded): Column definition for the custom table.
2. [Row](#) (min: 0, max: unbounded): Row definition for the custom table.

Attributes

Name	Type	Description	Required
Id	String	Identifier for the custom table.	Yes

See Also

[Wix Schema](#)

Data Element

Description

Data item for a row of a Custom Table

Windows Installer references

None

Parents

[Row](#)

Inner Text (xs:string)

Element value is data the data value

Children

None

Attributes

Name	Type	Description	Required
Column	String		Yes

See Also

[Wix Schema](#)

DeleteServices Element

Description

Stops a service and removes its registration from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Dependency Element

Description

Declares a dependency on another merge module.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
RequiredId	String	Identifier of the merge module required by the merge module.	Yes
RequiredLanguage	Integer	Numeric language ID of the merge module in RequiredID.	Yes
RequiredVersion	String	Version of the merge module in RequiredID.	

See Also

[Wix Schema](#)

Dialog Element

Description

None

Windows Installer references

[Control Table](#), [ComboBox Table](#), [Dialog Table](#), [ListBox Table](#),
[ListView Table](#), [RadioButton Table](#)

Parents

[UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Control](#) (min: 0, max: unbounded): Control elements belonging to this dialog

Attributes

Name	Type	Description	Required
Id	String		Yes
CustomPalette	YesNoType		
ErrorDialog	YesNoType		
Height	Integer	in dialog units	Yes
Hidden	YesNoType		
KeepModeless	YesNoType		
LeftScroll	YesNoType		
Modeless	YesNoType		
NoMinimize	YesNoType		
RightAligned	YesNoType		
RightToLeft	YesNoType		
SystemModal	YesNoType		

Title	String		
TrackDiskSpace	YesNoType		
Width	Integer	in dialog units	Yes
X	Integer	in %, defaults to centered on screen (50)	
Y	Integer	in %, defaults to centered on screen (50)	

See Also

[Wix Schema](#)

Version 2.0.4820.0

DialogRef Element

Description

Reference to a Dialog. This will cause the entire referenced section's contents to be included in the installer database.

Windows Installer references

None

Parents

[UI](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Dialog to reference.	Yes

See Also

[Wix Schema](#), [Dialog](#)

DigitalCertificate Element

Description

Adds a digital certificate.

Windows Installer references

[MsiDigitalCertificate Table](#)

Parents

[DigitalSignature](#), [PatchCertificates](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for a certificate file.	Yes
SourceFile	String	The path to the certificate file.	Yes

See Also

[Wix Schema](#)

DigitalSignature Element

Description

Adds a digital signature

Windows Installer references

[MsiDigitalSignature Table](#)

Parents

[Media](#)

Inner Text (xs:string)

Element value can be hex-encoded hash value

Children

Choice of elements (min: 0, max: unbounded)

- [DigitalCertificate](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
SourceFile	String	The path to signature's optional hash file.	

See Also

[Wix Schema](#)

Directory Element

Description

Directory layout for the product. Also specifies the mappings between source and target directories.

Windows Installer references

[Directory Table](#)

Parents

[Directory](#), [DirectoryRef](#), [Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Component](#) (min: 0, max: unbounded)
- [Directory](#) (min: 0, max: unbounded)
- [Merge](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	This value is the unique identifier of the directory entry.	Yes
FileSource	String	Used to set the file system source for this directory's child elements.	
LongName	LongFileNameType	Set this value to the non-8.3 name for the directory. This attribute cannot be specified unless the Name attribute is used to set	

		the short name for this directory.
LongSource	LongFileNameType	Set this value to the non-8.3 name of the directory on the source media for systems supporting long names. This attribute cannot be specified unless the SourceName attribute sets the short source name for this directory.
Name	String	The 8.3 name of the directory. Do not specify this attribute (or the LongName attribute) if this directory represents the same directory as the parent (see the Windows Installer SDK's Directory table topic for more information about the "." operator).
SourceName	ShortFileNameType	The 8.3 name of the directory on the source media. If this attribute is not specified, the Windows Installer will default to the Name attribute.
src	String	This attribute has been deprecated; please use the FileSource attribute instead.

See Also

[Wix Schema](#), [DirectoryRef](#)

Version 2.0.4820.0

DirectoryRef Element

Description

Create a reference to a Directory element in another Fragment.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Component](#) (min: 0, max: unbounded)
- [Directory](#) (min: 0, max: unbounded)
- [Merge](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Directory element to reference.	Yes
FileSource	String	Used to set the file system source for this directory ref's child elements.	
src	String	This attribute has been deprecated; please use the FileSource attribute instead.	

See Also

[Wix Schema](#), [Directory](#)

DirectorySearch Element

Description

Searches for directory and assigns to value of parent Property.

Windows Installer references

[DrLocator Table](#), [Signature Table](#)

Parents

[ComplianceCheck](#), [ComplianceDrive](#), [ComponentSearch](#),
[DirectorySearch](#), [DirectorySearchRef](#), [IniFileSearch](#), [Property](#),
[RegistrySearch](#)

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [DirectorySearch](#) (min: 0, max: 1)
- [DirectorySearchRef](#) (min: 0, max: 1)
- [FileSearch](#) (min: 0, max: 1)
- [FileSearchRef](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String	External key into Signature table. If not in Signature table, search is for a directory defined with DirectorySearch elements.	Yes
Depth	Integer	Depth below the path that the installer searches for the file or directory specified by the search.	
Path	String	Path on the user's system. Either absolute, or relative to containing directories.	

See Also

[Wix Schema](#), [ComponentSearch](#), [IniFileSearch](#), [RegistrySearch](#)

Version 2.0.4820.0

DirectorySearchRef Element

Description

References an existing DirectorySearch element.

Windows Installer references

None

Parents

[ComplianceDrive](#), [ComponentSearch](#), [DirectorySearch](#), [DirectorySearchRef](#), [IniFileSearch](#), [Property](#), [RegistrySearch](#)

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [DirectorySearch](#) (min: 0, max: 1)
- [DirectorySearchRef](#) (min: 0, max: 1)
- [FileSearch](#) (min: 0, max: 1)
- [FileSearchRef](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String	Id of the search being referred to.	Yes
Parent	String	This attribute is the signature of the parent directory of the file or directory in the Signature_ column. If this field is null, and the Path column does not expand to a full path, then all the fixed drives of the user's system are searched by using the Path. This field is a key into one of the following tables: the RegLocator, the IniLocator, the CompLocator, or the DrLocator tables.	

Path	String	Path on the user's system. Either absolute, or relative to containing directories.
------	--------	--

See Also

[Wix Schema](#), [ComponentSearch](#), [IniFileSearch](#), [RegistrySearch](#)

Version 2.0.4820.0

DisableRollback Element

Description

Disables rollback for the remainder of the installation. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

DuplicateFiles Element

Description

Duplicates files installed by the InstallFiles action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

EnsureTable Element

Description

Use this element to ensure that a table appears in the installer database, even if its empty.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The name of the table.	Yes

Remarks

This element is particularly useful for two problems that may occur while merging merge modules:

1. The first likely problem is that in order to properly merge you need to have certain tables present prior to merging. Using this element is one way to ensure those tables are present prior to the merging.
2. The other common problem is that a merge module has incorrect validation information about some tables. By ensuring these tables prior to merging, you can avoid this problem because the correct validation information will go into the installer database before the merge module has a chance to set it incorrectly.

See Also

Wix Schema

Version 2.0.4820.0

Environment Element

Description

Environment variables added or removed for parent Component

Windows Installer references

[Environment Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for environment entry.	Yes
Action	Enumeration	Specifies whether the environmental variable should be created, set or removed when the parent component is installed. This attribute's value should be one of the following: <i>create</i> Creates the environment variable if it does not exist, then set it during installation. This has no effect on the value of the environment variable if it already exists.	

set

Creates the environment variable if it does not exist, and then set it during installation. If the environment variable exists, set it during the installation.

remove

Removes the environment variable during an installation. The installer only removes an environment variable during an installation if the name and value of the variable match the entries in the Name and Value fields of the Environment table. If you want to remove an environment variable, regardless of its value, use the '!' syntax, and leave the Value field empty.

Name	String	Name of the environment variable.	Yes
Part	Enumeration	This attribute's value should be one of the following: <i>all</i> This value is the entire environmental variable. <i>first</i> This value is prefixed. <i>last</i> This value is appended.	
Permanent	YesNoType	Specifies that the environment variable should not be removed	

		on uninstall.	
Separator	String	Optional attribute to change the separator used between values. By default a semi-colon is used.	
System	YesNoType	Specifies that the environment variable should be added to the system environment space. The default is 'no' which indicates the environment variable is added to the user environment space.	
Value	String	Value to set into the environment variable.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

Error Element

Description

None

Windows Installer references

[Error Table](#)

Parents

[UI](#)

Inner Text (xs:string)

Element value is Message, use CDATA if message contains delimiter characters

Children

None

Attributes

Name	Type	Description	Required
Id	Integer	Number of the error for which a message is being provided. See MSI SDK for error definitions.	

See Also

[Wix Schema](#)

Exclusion Element

Description

Declares a merge module with which this merge module is incompatible.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
ExcludedId	String	Identifier of the merge module that is incompatible.	Yes
ExcludedMaxVersion	String	Maximum version excluded from a range. If not set, all versions after min are excluded. If neither max nor min, no exclusion based on version.	
ExcludedMinVersion	String	Minimum version excluded from a range. If not set, all versions before max are excluded. If neither max nor min, no exclusion	

		based on version.
ExcludeExceptLanguage	Integer	Numeric language ID of the merge module in ExcludedID. All except this language will be excluded. Only one of ExcludeExceptLanguage and ExcludeLanguage may be specified.
ExcludeLanguage	Integer	Numeric language ID of the merge module in ExcludedID. The specified language will be excluded. Only one of ExcludeExceptLanguage and ExcludeLanguage may be specified.

See Also

[Wix Schema](#)

ExecuteAction Element

Description

Initiates the execution sequence. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminUISequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Extension Element

Description

Extension for a Component

Windows Installer references

[MIME Table](#), [Verb Table](#), [Registry Table](#)

Parents

[Component](#), [Include](#), [ProgId](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [MIME](#) (min: 0, max: unbounded)
- [Verb](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	This is simply the file extension, like "doc" or "xml". Do not include the preceding period.	Yes
Advertise	YesNoType	Whether this extension is to be advertised. The default is "no".	
ContentType	String	The MIME type that is to be written.	

See Also

[Wix Schema](#)

ExternalFile Element

Description

Contains information about specific files that are not part of a regular target image.

Windows Installer references

None

Parents

[Family](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ProtectRange](#) (min: 1, max: unbounded)
2. [SymbolPath](#) (min: 1, max: unbounded)
3. Choice of elements (min: 0, max: unbounded)
 - [IgnoreRange](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
File	String	Foreign key into the File table.	Yes
Order	Int	Specifies the order of the external files to use when creating the patch.	Yes
Source	String	Full path of the external file.	
src	String	This attribute has been deprecated; please use the Source attribute instead.	

See Also

[Wix Schema](#)

Family Element

Description

Group of one or more upgraded images of a product.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [UpgradeImage](#) (min: 1, max: unbounded)
2. Choice of elements (min: 0, max: unbounded)
 - [ExternalFile](#) (min: 0, max: unbounded)
 - [ProtectFile](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
DiskId	Int	Entered into the DiskId field of the new Media table record.	
DiskPrompt	String	Value to display in the "[1]" of the DiskPrompt Property. Using this attribute will require you to define a DiskPrompt Property.	
MediaSrcProp	String	Entered into the Source field of the new Media table entry of the upgraded image.	
Name	String	Identifier for the family.	Yes
SequenceStart	Int	Sequence number for the starting file.	

VolumeLabel	String	Entered into the VolumeLabel field of the new Media table record.
-------------	--------	---

See Also

[Wix Schema](#)

Version 2.0.4820.0

Feature Element

Description

A feature for the Feature table. Features are the smallest installable unit. See msi.chm for more detailed information on the myriad installation options for a feature.

Windows Installer references

[Feature Table](#)

Parents

[Feature](#), [FeatureRef](#), [Fragment](#), [Include](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [ComponentGroupRef](#) (min: 0, max: unbounded)
- [ComponentRef](#) (min: 0, max: unbounded)
- [Condition](#) (min: 0, max: unbounded)
- [Feature](#) (min: 0, max: unbounded)
- [FeatureRef](#) (min: 0, max: unbounded)
- [MergeRef](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description
Id	String	Unique identifier of the feature.
Absent	Enumeration	This attribute determines if a user option to set a feature to absent is allowed. This attribute's value should be one of the following: <i>allow</i> Allows the user interface to change the feature state to <i>Absent</i> .

disallow

Prevents the user interface from displaying the option to change the feature setting by setting the `msidbFeatureAttributesUIDisallow` attribute. This will force the feature to its installation state, whether or not it is visible in the UI.

AllowAdvertise	Enumeration	This attribute determines the possible values for this feature. This attribute's value is one of the following: <i>no</i> Prevents this feature from being advertised by setting the <code>msidbFeatureAttributesDisallowAdvertise</code> attribute. <i>system</i> Prevents advertising for this feature if the operating system shell does not support the Windows Installer descriptor <code>msidbFeatureAttributesNoUI</code> attribute. <i>yes</i> Allows the feature to be advertised.
ConfigurableDirectory	String	Specify the Id of a Directory that can be changed by the user at installation time. This must be a public property and therefore the name must be uppercase.
Description	String	Longer string of text describing the feature. The localizable string is displayed by the Selection Dialog.
Display	String	Determines the initial display of the feature tree. This attribute's value is one of the following: <i>collapse</i> Initially shows the feature collapsed.

default value.

expand

Initially shows the feature ex

hidden

Prevents the feature from dis
interface.

<an explicit integer value>

For advanced users only, it is
set the integer value of the d
appear in the Feature row.

InstallDefault	Enumeration	This attribute determines the default location of a feature. This attribute is specified if the value of the FollowParent attribute is 'yes' since that would ask the installer to follow the parent install simultaneously favor a particular location just for this feature. This attribute can be one of the following: <i>followParent</i> Forces the feature to follow the parent's state as its parent feature. <i>local</i> Favors installing this feature locally. This is the msidbFeatureAttributesFavorLocal attribute. <i>source</i> Favors running this feature from the source. This is the msidbFeatureAttributesFavorSource attribute.
Level	Integer	Sets the install level of this feature. A value of 0 disables the feature. Processing the feature can modify the level value (this is the msidbFeatureAttributesCondition child element).
Title	String	Short string of text identifying the feature. This is listed as an item by the Selecti

the Selection Dialog.

TypicalDefault	Enumeration	This attribute determines the default of the feature. This attribute's value is one of the following:
		<i>advertise</i> Sets the feature to be advertised. This value cannot be set if the AllowAdvertise attribute is 'no' or 'ask' to ask the installer to disallow this feature while at the selection dialog. <i>install</i> Sets the feature to the default installation option.

Any attribute namespace='###other' processContents='lax'

See Also

[Wix Schema](#), [FeatureRef](#)

FeatureRef Element

Description

Create a reference to a Feature element in another Fragment.

Windows Installer references

None

Parents

[Feature](#), [FeatureRef](#), [Fragment](#), [Include](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [ComponentGroupRef](#) (min: 0, max: unbounded)
- [ComponentRef](#) (min: 0, max: unbounded)
- [Feature](#) (min: 0, max: unbounded)
- [FeatureRef](#) (min: 0, max: unbounded)
- [MergeRef](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Feature element to reference.	Yes

See Also

[Wix Schema](#), [Feature](#)

File Element

Description

File specification for File table, must be child node of Component

Windows Installer references

[File Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [AppId](#) (min: 0, max: unbounded)
- [AssemblyName](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [CopyFile](#) (min: 0, max: unbounded): to DuplicateFile table
- [ODBCDriver](#) (min: 0, max: unbounded)
- [ODBCTranslator](#) (min: 0, max: unbounded)
- [Patch](#) (min: 0, max: unbounded): to Patch table
- [PerfCounter](#) (min: 0, max: unbounded)
- [Permission](#) (min: 0, max: unbounded)
- [Shortcut](#) (min: 0, max: unbounded): Target is preset to this file
- [TypeLib](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)
- [HelpCollection](#)
- [HelpFile](#)
- [NativeImage](#)
- [SnapIn](#)

Attributes



Name	Type	Description	Required
Id	String		Yes
Assembly	Enumeration	<p>Specifies if this File is a Win32 Assembly or .NET Assembly; the default is neither. If the value is '.net' or 'win32', this file must also be the key path of the Component. This attribute's value should be one of the following:</p> <p><i>.net</i></p> <p><i>no</i></p> <p><i>win32</i></p>	
AssemblyApplication	String	<p>Specifies the file identifier of the application file. This assembly will be isolated to the same directory as the application file. If this attribute is absent, the assembly will be installed to the Global Assembly Cache. This</p>	

		attribute may only be specified if the Assembly attribute is set to '.net' or 'win32'.
AssemblyManifest	String	Specifies the file identifier of the manifest file that describes this assembly. The manifest file should be in the same component as the assembly it describes. This attribute may only be specified if the Assembly attribute is set to '.net' or 'win32'.
BindPath	String	generates BindImage table row, value may be empty string
Checksum	YesNoType	This attribute should be set to "yes" for every executable file in the installation that has a valid checksum

stored in the Portable Executable (PE) file header. Only those files that have this attribute set will be verified for valid checksum during a reinstall.

CompanionFile

String

Set this attribute to make this file a companion child of another file. The installation state of a companion file depends not on its own file versioning information, but on the versioning of its companion parent. A file that is the key path for its component can not be a companion file (that means this attribute cannot be set if KeyPath="yes")

for this file).
The Version attribute cannot be set along with this attribute since companion files are not installed based on their own version.

Compressed	YesNoDefaultType	Sets the file's source type compression. A setting of "yes" or "no" will override the setting in the Word Count Summary Property.
DefaultLanguage	String	This is the default language of this file. The linker will replace this value from the value in the file if the suppress files option is not used.
DefaultSize	Integer	This is the default size of this file. The linker will replace this value from the value in the file if the suppress

		files option is not used.
DefaultVersion	String	This is the default version of this file. The linker will replace this value from the value in the file if the suppress files option is not used.
DiskId	String	Specifies the Media this File should be sourced on. This attribute must be set on this File element or its parent Component.
FontTitle	String	generates entries in Font table with the FontTitle
Hidden	YesNoType	Set to yes in order to have the file's hidden attribute set when it is installed on the target machine.
KeyPath	YesNoType	Set yes to force this File to be key path for parent Component.

LongName	LongFileNameType	Long file name; set this attribute if preferred name is not in 8.3 format.	
Name	ShortFileNameType	File name of the file in 8.3 format, required for backwards compatibility.	Yes
PatchGroup	Integer	This attribute must be set for patch-added files. Each patch should be assigned a different patch group number. Patch groups numbers must be greater 0 and should be assigned consecutively. For example, the first patch should use PatchGroup='1', the second patch will have PatchGroup='2', etc...	
ProcessorArchitecture Enumeration		Specifies the architecture for this assembly. This attribute should only be used on .NET	

Assemblies for the CLR 2.0. This attribute's value should be one of the following:
msil
x86
x64
ia64

ReadOnly	YesNoType	Set to yes in order to have the file's read-only attribute set when it is installed on the target machine.
SelfRegCost	Integer	generates SelfReg table row
Source	String	Specifies the path to the File in the build process. This attribute must be set if no source information can be gathered from parent directories.
src	String	This attribute has been deprecated; please use the Source attribute

		instead.
System	YesNoType	Set to yes in order to have the file's system attribute set when it is installed on the target machine.
TrueType	YesNoType	generates entries in Font table with no FontTitle
Vital	YesNoType	If a file is vital, then installation cannot proceed unless the file is successfully installed. The user will have no option to ignore an error installing this file. If an error occurs, they can merely retry to install the file or abort the installation.

Any attribute namespace='###other' processContents='lax'

See Also

[Wix Schema](#)

FileCost Element

Description

Initiates dynamic costing of standard installation actions. Any standard or custom actions that affect costing should be sequenced before the CostInitialize action. Call the FileCost action immediately following the CostInitialize action. Then call the CostFinalize action following the FileCost action to make all final cost calculations available to the installer through the Component table. The CostInitialize action must be executed before the FileCost action. The installer then determines the disk-space cost of every file in the File table, on a per-component basis, taking both volume clustering and the presence of existing files that may need to be overwritten into account. All actions that consume or release disk space are also considered. If an existing file is found, a file version check is performed to determine whether the new file actually needs to be installed or not. If the existing file is of an equal or greater version number, the existing file is not overwritten and no disk-space cost is incurred. In all cases, the installer uses the results of version number checking to set the installation state of each file. The FileCost action initializes cost calculation with the installer. Actual dynamic costing does not occur until the CostFinalize action is executed. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#), [CostInitialize](#), [CostFinalize](#)

Version 2.0.4820.0

FileSearch Element

Description

Searches for file and assigns to fullpath value of parent Property

Windows Installer references

[DrLocator Table](#), [Signature Table](#)

Parents

[ComplianceCheck](#), [ComponentSearch](#), [DirectorySearch](#),
[DirectorySearchRef](#), [IniFileSearch](#), [Property](#), [RegistrySearch](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Specify the Id when you want to find the path to a file. Leave the Id absent if you want to find the parent directory of a file.	
Languages	String	The languages supported by the file.	
LongName	LongFileNameType	Long file name; set this attribute if preferred name is not in 8.3 format. Either this attribute or the Name attribute is required. When using only the LongName attribute, ICE03 should be ignored	

for the Signature table's
FileName column.

MaxDate	DateTime	The maximum modification date and time of the file. Formatted as YYYY-MM-DDTHH:mm:ss, where YYYY is the year, MM is month, DD is day, 'T' is literal, HH is hour, mm is minute and ss is second.
MaxSize	Int	The maximum size of the file.
MaxVersion	String	The maximum version of the file.
MinDate	DateTime	The minimum modification date and time of the file. Formatted as YYYY-MM-DDTHH:mm:ss, where YYYY is the year, MM is month, DD is day, 'T' is literal, HH is hour, mm is minute and ss is second.
MinSize	Int	The minimum size of the file.
MinVersion	String	The minimum version of the file.
Name	ShortFileNameType	File name of the file in 8.3 format. Please note that due to a Windows Installer bug, this attribute is not required if the preferred file name is not in 8.3 format. This attribute should only be set if the file to find is in

8.3 format. Either this attribute or the LongName is required.

See Also

[Wix Schema](#), [ComponentSearch](#), [DirectorySearch](#), [DirectorySearchRef](#), [FileSearchRef](#), [IniFileSearch](#), [RegistrySearch](#)

Version 2.0.4820.0

FileSearchRef Element

Description

References an existing FileSearch element.

Windows Installer references

None

Parents

[ComponentSearch](#), [DirectorySearch](#), [DirectorySearchRef](#),
[IniFileSearch](#), [RegistrySearch](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Specify the Id to the FileSearch to reference.	Yes

See Also

[Wix Schema](#), [FileSearch](#)

FileShare Element

Description

Creates a file share out of the component's directory.

Windows Installer references

None

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Permission](#) (min: 1, max: unbounded): ACL permission

Attributes

Name	Type	Description	Required
Id	String	Identifier for the file share (primary key).	Yes
Description	String	Description of the file share.	
Name	String	Name of the file share.	Yes

See Also

[Wix Schema](#)

FileTypeMask Element

Description

FileType data for class Id registration.

Windows Installer references

None

Parents

[Class](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Mask	HexType	Hex value that is AND'd against the bytes in the file at Offset.	Yes
Offset	Integer	Offset into file. If positive, offset is from the beginning; if negative, offset is from the end.	Yes
Value	HexType	If the result of the AND'ing of Mask with the bytes in the file is Value, the file is a match for this File Type.	Yes

See Also

[Wix Schema](#)

FindRelatedProducts Element

Description

Runs through each record of the Upgrade table in sequence and compares the upgrade code, product version, and language in each row to products installed on the system. When FindRelatedProducts detects a correspondence between the upgrade information and an installed product, it appends the product code to the property specified in the ActionProperty column of the UpgradeTable. The FindRelatedProducts action only runs the first time the product is installed. The FindRelatedProducts action does not run during maintenance mode or uninstallation. FindRelatedProducts should be authored into the InstallUISequence table and InstallExecuteSequence tables. The installer prevents FindRelatedProducts from running in InstallExecuteSequence if the action has already run in InstallUISequence. The FindRelatedProducts action must come before the MigrateFeatureStates action and the RemoveExistingProducts action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress [YesNoType](#) If yes, this action will not occur.

See Also

[Wix Schema](#), [Upgrade](#)

Version 2.0.4820.0

ForceReboot Element

Description

Prompts the user for a restart of the system during the installation. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Fragment Element

Description

The Fragment element is the building block of creating an installer database in WiX. Once defined, the Fragment becomes an immutable, atomic unit which can either be completely included or excluded from a product. The contents of a Fragment element can be linked into a product by utilizing one of the many *Ref elements. When linking in a Fragment, it will be necessary to link in all of its individual units. For instance, if a given Fragment contains two Component elements, you must link both under features using ComponentRef for each linked Component. Otherwise, you will get a linker warning and have a floating Component that does not appear under any Feature.

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [AppId](#) (min: 0, max: unbounded)
 - [Binary](#) (min: 0, max: unbounded)
 - [ComplianceCheck](#) (min: 0, max: unbounded)
 - [ComponentGroup](#) (min: 0, max: unbounded)
 - [Condition](#) (min: 0, max: unbounded)
 - [CustomAction](#) (min: 0, max: unbounded)
 - [CustomActionRef](#) (min: 0, max: unbounded)
 - [CustomTable](#) (min: 0, max: unbounded)

- [Directory](#) (min: 0, max: unbounded)
- [DirectoryRef](#) (min: 0, max: unbounded)
- [EnsureTable](#) (min: 0, max: unbounded)
- [Feature](#) (min: 0, max: unbounded)
- [FeatureRef](#) (min: 0, max: unbounded)
- [FragmentRef](#) (min: 0, max: unbounded)
- [Group](#) (min: 0, max: unbounded)
- [Icon](#) (min: 0, max: unbounded)
- [IgnoreModularization](#) (min: 0, max: unbounded)
- [Media](#) (min: 0, max: unbounded)
- [PatchCertificates](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [SqlDatabase](#) (min: 0, max: unbounded)
- [UI](#) (min: 0, max: unbounded)
- [UIRef](#) (min: 0, max: unbounded)
- [Upgrade](#) (min: 0, max: unbounded)
- [User](#) (min: 0, max: unbounded)
- [WebApplication](#) (min: 0, max: unbounded)
- [WebAppPool](#) (min: 0, max: unbounded)
- [WebDirProperties](#) (min: 0, max: unbounded)
- [WebLog](#) (min: 0, max: unbounded)
- [WebSite](#) (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 1. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- [Any Element namespace='###other' processContents='Lax'](#)
- [HelpCollectionRef](#)
- [HelpFilter](#)

Attributes

Name	Type	Description	Required
Id	String	Optional identifier for a Fragment. Should only be used if you plan to refer to this Fragment via a FragmentRef element elsewhere.	

See Also

[Wix Schema](#), [FragmentRef](#)

Version 2.0.4820.0

FragmentRef Element

Description

Reference to a Fragment. This will force the entire referenced Fragment's contents to be included in the installer database.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Fragment to reference.	Yes

See Also

[Wix Schema](#), [Fragment](#)

Group Element

Description

Group for all kinds of things

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Domain	String		
Name	String		Yes

See Also

[Wix Schema](#)

GroupRef Element

Description

Used to join a user to a group

Windows Installer references

None

Parents

[User](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes

See Also

[Wix Schema](#)

HTTPHeader Element

Description

Custom HTTP Header definition for IIS resources such as WebSite and WebVirtualDir.

Windows Installer references

None

Parents

[WebSite](#), [WebVirtualDir](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Name	String	Name of the custom HTTP Header.	Yes
Value	String	Value for the custom HTTP Header. This attribute may be set via a formatted Property (e.g. [MyProperty]).	

See Also

[Wix Schema](#)

Icon Element

Description

Icon used for Shortcut, ProgId, or Class elements (but not UI controls)

Windows Installer references

[Icon Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#), [Shortcut](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The Id cannot be longer than 55 characters. In order to prevent errors in cases where the Id is modularized, it should not be longer than 18 characters.	Yes
SourceFile	String	Path to the icon file.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

See Also

[Wix Schema](#)

IgnoreModularization Element

Description

Use this to Ignore Modularization of particular values. This feature is intended to be used in very rare situations. Before using this feature, contact your support alias to verify your use is supported.

Windows Installer references

None

Parents

[Fragment](#), [Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Name	String		Yes
Type	Enumeration	This attribute's value should be one of the following: <i>Action</i> <i>Property</i> <i>Directory</i>	

See Also

[Wix Schema](#)

IgnoreRange Element

Description

Specifies part of a file that is to be ignored during patching.

Windows Installer references

None

Parents

[ExternalFile](#), [TargetFile](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Length	Int	Length of the range.	Yes
Offset	Int	Offset of the start of the range.	Yes

See Also

[Wix Schema](#)

Include Element

Description

This is the top-level container element for every wxi file.

Windows Installer references

None

Parents

None

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [AppId](#) (min: 0, max: unbounded)
- [Binary](#) (min: 0, max: unbounded)
- [Category](#) (min: 0, max: unbounded)
- [Certificate](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [ComplianceCheck](#) (min: 0, max: unbounded)
- [Condition](#) (min: 0, max: unbounded)
- [CopyFile](#) (min: 0, max: unbounded)
- [CreateFolder](#) (min: 0, max: unbounded)
- [CustomAction](#) (min: 0, max: unbounded)
- [CustomActionRef](#) (min: 0, max: unbounded)
- [CustomTable](#) (min: 0, max: unbounded)
- [Directory](#) (min: 0, max: unbounded)
- [DirectoryRef](#) (min: 0, max: unbounded)
- [EnsureTable](#) (min: 0, max: unbounded)
- [Environment](#) (min: 0, max: unbounded)
- [Extension](#) (min: 0, max: unbounded)
- [Feature](#) (min: 0, max: unbounded)

- [FeatureRef](#) (min: 0, max: unbounded)
- [File](#) (min: 0, max: unbounded)
- [FileShare](#) (min: 0, max: unbounded)
- [FragmentRef](#) (min: 0, max: unbounded)
- [Group](#) (min: 0, max: unbounded)
- [Icon](#) (min: 0, max: unbounded)
- [IniFile](#) (min: 0, max: unbounded)
- [Interface](#) (min: 0, max: unbounded)
- [IsolateComponent](#) (min: 0, max: unbounded)
- [Media](#) (min: 0, max: unbounded)
- [ODBCDataSource](#) (min: 0, max: unbounded)
- [ODBCDriver](#) (min: 0, max: unbounded)
- [ODBCTranslator](#) (min: 0, max: unbounded)
- [ProgId](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [Registry](#) (min: 0, max: unbounded)
- [RemoveFile](#) (min: 0, max: unbounded)
- [RemoveFolder](#) (min: 0, max: unbounded)
- [ReserveCost](#) (min: 0, max: unbounded)
- [ServiceConfig](#) (min: 0, max: unbounded)
- [ServiceControl](#) (min: 0, max: unbounded)
- [ServiceInstall](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [Shortcut](#) (min: 0, max: unbounded)
- [SqlDatabase](#) (min: 0, max: unbounded)
- [SqlScript](#) (min: 0, max: unbounded)
- [SqlString](#) (min: 0, max: unbounded)
- [TypeLib](#) (min: 0, max: unbounded)
- [UI](#) (min: 0, max: unbounded)
- [UIRef](#) (min: 0, max: unbounded)
- [Upgrade](#) (min: 0, max: unbounded)
- [User](#) (min: 0, max: unbounded)

- [WebApplication](#) (min: 0, max: unbounded)
- [WebAppPool](#) (min: 0, max: unbounded)
- [WebDir](#) (min: 0, max: unbounded)
- [WebDirProperties](#) (min: 0, max: unbounded)
- [WebFilter](#) (min: 0, max: unbounded)
- [WebLog](#) (min: 0, max: unbounded)
- [WebProperty](#) (min: 0, max: unbounded)
- [WebServiceExtension](#) (min: 0, max: unbounded)
- [WebSite](#) (min: 0, max: unbounded)
- [WebVirtualDir](#) (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 1. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)

Attributes

None

See Also

[Wix Schema](#)

IniFile Element

Description

Adds or removes .ini file entries.

Windows Installer references

[IniFile Table](#), [RemoveIniFile Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for ini file.	Yes
Action	Enumeration	The type of modification to be made. This attribute's value should be one of the following: <i>addLine</i> Creates or updates an .ini entry. <i>addTag</i> Creates a new entry or appends a new comma-separated value to an existing entry. <i>createLine</i> Creates an .ini entry only if the entry does	Yes

no already exist.

removeLine

Removes an .ini entry.

removeTag

Removes a tag from an .ini entry.

Directory	String	Name of a property, the value of which is the full path of the folder containing the .ini file. Can be name of a directory in the Directory table, a property set by the AppSearch table, or any other property representing a full path.	
Key	String	The localizable .ini file key within the section.	Yes
LongName	LongFileNameType	Long file name; set this attribute if preferred name is not in 8.3 format.	
Name	ShortFileNameType	File name of the file in 8.3 format, required for backwards compatibility.	Yes
Section	String	The localizable .ini file section.	Yes
Value	String	The localizable value to be written or deleted. This attribute must be set if the Action attribute's value is "addLine", "addTag", or "createLine".	

See Also

[Wix Schema](#)

Version 2.0.4820.0

IniFileSearch Element

Description

Searches for file, directory or registry key and assigns to value of parent Property

Windows Installer references

[IniLocator Table](#), [Signature Table](#)

Parents

[ComplianceCheck](#), [Property](#)

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [DirectorySearch](#) (min: 0, max: 1)
- [DirectorySearchRef](#) (min: 0, max: 1)
- [FileSearch](#) (min: 0, max: 1)
- [FileSearchRef](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String	External key into the Signature table.	Yes
Field	Integer	The field in the .ini line. If field is Null or 0, the entire line is read.	
Key	String	The key value within the section.	Yes
LongName	LongFileNameType	Long file name; set this attribute if preferred name is not in 8.3 format.	
Name	ShortFileNameType	File name of the file in 8.3	

		format, required for backwards compatibility.	
Section	String	The section name within the .ini file.	Yes
Type	Enumeration	<p>Must be file if last child is FileSearch element and must be directory if last child is DirectorySearch element. This attribute's value should be one of the following:</p> <p><i>directory</i> A directory location.</p> <p><i>file</i> A file location.</p> <p><i>raw</i> A raw .ini value.</p>	

See Also

[Wix Schema](#), [ComponentSearch](#), [RegistrySearch](#)

InstallAdminPackage Element

Description

Copies the product database to the administrative installation point. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

InstallExecute Element

Description

Runs a script containing all operations spooled since either the start of the installation or the last InstallExecute action, or InstallExecuteAgain action. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

InstallExecuteAgain Element

Description

Runs a script containing all operations spooled since either the start of the installation or the last InstallExecute action, or InstallExecuteAgain action. Should only be used after InstallExecute. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

InstallExecuteSequence Element

Description

None

Windows Installer references

[InstallExecuteSequence Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [AllocateRegistrySpace](#) (min: 0, max: unbounded): Ensures the needed amount of space exists in the registry.
- [AppSearch](#) (min: 0, max: unbounded): Uses file signatures to search for existing versions of products.
- [BindImage](#) (min: 0, max: unbounded): Binds each executable or DLL that must be bound to the DLLs imported by it.
- [CCPSearch](#) (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- [CostFinalize](#) (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- [CostInitialize](#) (min: 0, max: unbounded): Initiates the internal installation costing process.
- [CreateFolders](#) (min: 0, max: unbounded): Creates empty folders for components that are set to be installed.
- [CreateShortcuts](#) (min: 0, max: unbounded): Manages the creation of shortcuts.
- [Custom](#) (min: 0, max: unbounded): Use to sequence a custom action.

- [DeleteServices](#) (min: 0, max: unbounded): Stops a service and removes its registration from the system.
- [DisableRollback](#) (min: 0, max: unbounded): Disables rollback for the remainder of the installation.
- [DuplicateFiles](#) (min: 0, max: unbounded): Duplicates files installed by the InstallFiles action.
- [FileCost](#) (min: 0, max: unbounded): Initiates dynamic costing of standard installation actions.
- [FindRelatedProducts](#) (min: 0, max: unbounded): Runs through each record of the Upgrade table in sequence and compares the upgrade code, product version, and language in each row to products installed on the system.
- [ForceReboot](#) (min: 0, max: unbounded): Prompts the user for a restart of the system during the installation. Not fixed sequence.
- [InstallExecute](#) (min: 0, max: unbounded): Runs a script containing all operations spooled since either the start of the installation or the last InstallExecute action, or InstallExecuteAgain action.
- [InstallExecuteAgain](#) (min: 0, max: unbounded): Runs a script containing all operations spooled since either the start of the installation or the last InstallExecute action, or InstallExecuteAgain action.
- [InstallFiles](#) (min: 0, max: unbounded): Copies files specified in the File table from the source directory to the destination directory.
- [InstallFinalize](#) (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- [InstallInitialize](#) (min: 0, max: unbounded): Marks the beginning of a sequence of actions that change the system.
- [InstallODBC](#) (min: 0, max: unbounded): Installs the drivers, translators, and data sources in the ODBCDriver table, ODBCTranslator table, and ODBCDataSource table.
- [InstallServices](#) (min: 0, max: unbounded): Registers a service for the system.
- [InstallValidate](#) (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.
- [IsolateComponents](#) (min: 0, max: unbounded): Installs a copy of a component (commonly a shared DLL) into a private location for use

by a specific application (typically an .exe).

- [LaunchConditions](#) (min: 0, max: unbounded): Queries the LaunchCondition table and evaluates each conditional statement recorded there.
- [MigrateFeatureStates](#) (min: 0, max: unbounded): Used for upgrading or installing over an existing application.
- [MoveFiles](#) (min: 0, max: unbounded): Locates existing files on the system and moves or copies those files to a new location.
- [MsiPublishAssemblies](#) (min: 0, max: unbounded): Manages the advertisement of CLR and Win32 assemblies.
- [MsiUnpublishAssemblies](#) (min: 0, max: unbounded): Manages the unadvertisement of CLR and Win32 assemblies that are being removed.
- [PatchFiles](#) (min: 0, max: unbounded): Queries the Patch table to determine which patches are to be applied.
- [ProcessComponents](#) (min: 0, max: unbounded): Registers and unregisters components, their key paths, and the component clients.
- [PublishComponents](#) (min: 0, max: unbounded): Manages the advertisement of the components from the PublishComponent table.
- [PublishFeatures](#) (min: 0, max: unbounded): Writes each feature's state into the system registry.
- [PublishProduct](#) (min: 0, max: unbounded): Manages the advertisement of the product information with the system.
- [RegisterClassInfo](#) (min: 0, max: unbounded): Manages the registration of COM class information with the system.
- [RegisterComPlus](#) (min: 0, max: unbounded): Registers COM+ applications.
- [RegisterExtensionInfo](#) (min: 0, max: unbounded): Manages the registration of extension related information with the system.
- [RegisterFonts](#) (min: 0, max: unbounded): Registers installed fonts with the system.
- [RegisterMIMEInfo](#) (min: 0, max: unbounded): Registers MIME-related registry information with the system.
- [RegisterProduct](#) (min: 0, max: unbounded): Registers the product information with the installer.
- [RegisterProgIdInfo](#) (min: 0, max: unbounded): Manages the

registration of OLE ProgId information with the system.

- [RegisterTypeLibraries](#) (min: 0, max: unbounded): Registers type libraries with the system.
- [RegisterUser](#) (min: 0, max: unbounded): Registers the user information with the installer to identify the user of a product.
- [RemoveDuplicateFiles](#) (min: 0, max: unbounded): Deletes files installed by the DuplicateFiles action.
- [RemoveEnvironmentStrings](#) (min: 0, max: unbounded): Modifies the values of environment variables.
- [RemoveExistingProducts](#) (min: 0, max: unbounded): Goes through the product codes listed in the ActionProperty column of the Upgrade table and removes the products in sequence.
- [RemoveFiles](#) (min: 0, max: unbounded): Removes files previously installed by the InstallFiles action.
- [RemoveFolders](#) (min: 0, max: unbounded): Removes any folders linked to components set to be removed or run from source.
- [RemoveIniValues](#) (min: 0, max: unbounded): Removes .ini file information specified for removal in the RemoveIniFile table if the component is set to be installed locally or run from source.
- [RemoveODBC](#) (min: 0, max: unbounded): Removes the data sources, translators, and drivers listed for removal during the installation.
- [RemoveRegistryValues](#) (min: 0, max: unbounded): Removes a registry value that has been authored into the registry table if the associated component was installed locally or as run from source, and is now set to be uninstalled.
- [RemoveShortcuts](#) (min: 0, max: unbounded): Manages the removal of an advertised shortcut whose feature is selected for uninstallation or a nonadvertised shortcut whose component is selected for uninstallation.
- [ResolveSource](#) (min: 0, max: unbounded): Determines the location of the source and sets the SourceDir property if the source has not been resolved yet. Not fixed sequence.
- [RMCCPSearch](#) (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.

- [ScheduleReboot](#) (min: 0, max: unbounded): Prompts the user to restart the system at the end of installation. Not fixed sequence.
- [SelfRegModules](#) (min: 0, max: unbounded): Processes all modules listed in the SelfReg table and registers all installed modules with the system.
- [SelfUnregModules](#) (min: 0, max: unbounded): Unregisters all modules listed in the SelfReg table that are scheduled to be uninstalled.
- [SetODBCFolders](#) (min: 0, max: unbounded): Checks for existing ODBC drivers and sets the target directory for each new driver to the location of an existing driver.
- [StartServices](#) (min: 0, max: unbounded): Starts system services.
- [StopServices](#) (min: 0, max: unbounded): Stops system services.
- [UnpublishComponents](#) (min: 0, max: unbounded): Manages the unadvertisement of components listed in the PublishComponent table.
- [UnpublishFeatures](#) (min: 0, max: unbounded): Removes selection-state and feature-component mapping information from the registry.
- [UnregisterClassInfo](#) (min: 0, max: unbounded): Manages the removal of COM class information from the system registry.
- [UnregisterComPlus](#) (min: 0, max: unbounded): Removes COM+ applications from the registry.
- [UnregisterExtensionInfo](#) (min: 0, max: unbounded): Manages the removal of extension-related information from the system registry.
- [UnregisterFonts](#) (min: 0, max: unbounded): Removes registration information about installed fonts from the system.
- [UnregisterMIMEInfo](#) (min: 0, max: unbounded): Unregisters MIME-related registry information from the system.
- [UnregisterProgIdInfo](#) (min: 0, max: unbounded): Manages the unregistration of OLE ProgId information with the system.
- [UnregisterTypeLibraries](#) (min: 0, max: unbounded): Unregisters type libraries from the system.
- [ValidateProductID](#) (min: 0, max: unbounded): Sets the ProductID property to the full product identifier.
- [WriteEnvironmentStrings](#) (min: 0, max: unbounded): Modifies the values of environment variables.

- [WriteIniValues](#) (min: 0, max: unbounded): Writes the .ini file information that the application needs written to its .ini files.
- [WriteRegistryValues](#) (min: 0, max: unbounded): Sets up an application's registry information.

Attributes

None

See Also

[Wix Schema](#)

Version 2.0.4820.0

InstallFiles Element

Description

Copies files specified in the File table from the source directory to the destination directory. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

InstallFinalize Element

Description

Marks the end of a sequence of actions that change the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#), [InstallInitialize](#)

InstallInitialize Element

Description

Marks the beginning of a sequence of actions that change the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#), [InstallFinalize](#)

InstallODBC Element

Description

Installs the drivers, translators, and data sources in the ODBCDriver table, ODBCTranslator table, and ODBCDataSource table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

InstallServices Element

Description

Registers a service for the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

InstallUISequence Element

Description

None

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#), [UI](#)

Inner Text (xs:string)

This element may have inner text.

Children

Choice of elements (min: 0, max: unbounded)

- [AppSearch](#) (min: 0, max: unbounded): Uses file signatures to search for existing versions of products.
- [CCPSearch](#) (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- [CostFinalize](#) (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- [CostInitialize](#) (min: 0, max: unbounded): Initiates the internal installation costing process.
- [Custom](#) (min: 0, max: unbounded): Use to sequence a custom action.
- [ExecuteAction](#) (min: 0, max: unbounded): Initiates the execution sequence.
- [FileCost](#) (min: 0, max: unbounded): Initiates dynamic costing of standard installation actions.
- [FindRelatedProducts](#) (min: 0, max: unbounded): Runs through each record of the Upgrade table in sequence and compares the upgrade code, product version, and language in each row to products installed on the system.
- [IsolateComponents](#) (min: 0, max: unbounded): Installs a copy of a

component (commonly a shared DLL) into a private location for use by a specific application (typically an .exe).

- [LaunchConditions](#) (min: 0, max: unbounded): Queries the LaunchCondition table and evaluates each conditional statement recorded there.
- [MigrateFeatureStates](#) (min: 0, max: unbounded): Used for upgrading or installing over an existing application.
- [ResolveSource](#) (min: 0, max: unbounded): Determines the location of the source and sets the SourceDir property if the source has not been resolved yet.
- [RMCCPSearch](#) (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- [ScheduleReboot](#) (min: 0, max: unbounded): Prompts the user to restart the system at the end of installation. Not fixed sequence.
- [Show](#) (min: 0, max: unbounded)
- [ValidateProductID](#) (min: 0, max: unbounded): Sets the ProductID property to the full product identifier.

Attributes

None

See Also

[Wix Schema](#)

InstallValidate Element

Description

Verifies that all costed volumes have enough space for the installation. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Interface Element

Description

COM Interface registration for parent Typelib.

Windows Installer references

[Registry Table](#)

Parents

[Class](#), [Component](#), [Include](#), [TypeLib](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Uuid	GUID identifier for COM Interface.	Yes
Name	String	Name for COM Interface.	Yes
NumMethods	Integer	Number of methods implemented on COM Interface.	
ProxyStubClassId	Uuid	GUID CLSID for proxy stub to COM Interface.	
ProxyStubClassId32	Uuid	GUID CLSID for 32-bit proxy stub to COM Interface.	
Versioned	YesNoType	Determines whether a Typelib version entry should be created with the other COM Interface registry keys. Default is	

'yes'.

See Also

[Wix Schema](#)

Version 2.0.4820.0

IsolateComponent Element

Description

Shared Component to be privately replicated in folder of parent Component

Windows Installer references

[IsolateComponent Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Shared	String	Shared Component for this application Component	Yes

See Also

[Wix Schema](#)

IsolateComponents Element

Description

Installs a copy of a component (commonly a shared DLL) into a private location for use by a specific application (typically an .exe). This isolates the application from other copies of the component that may be installed to a shared location on the computer. The action refers to each record of the IsolatedComponent table and associates the files of the component listed in the Component_Shared field with the component listed in the Component_Application field. The installer installs the files of Component_Shared into the same directory as Component_Application. The installer generates a file in this directory, zero bytes in length, having the short filename name of the key file for Component_Application (typically this is the same file name as the .exe) appended with .local. The IsolatedComponent action does not affect the installation of Component_Application. Uninstalling Component_Application also removes the Component_Shared files and the .local file from the directory. The IsolateComponents action can be used only in the InstallUISequence table and the InstallExecuteSequence table. This action must come after the CostInitialize action and before the CostFinalize action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
------	------	-------------	----------

Sequence Integer	A value used to indicate the position of this action in a sequence.
------------------	---

Suppress	YesNoType	If yes, this action will not occur.
----------	---------------------------	-------------------------------------

See Also

[Wix Schema](#), [IsolateComponent](#)

Version 2.0.4820.0

LaunchConditions Element

Description

Queries the LaunchCondition table and evaluates each conditional statement recorded there. If any of these conditional statements fail, an error message is displayed to the user and the installation is terminated. The LaunchConditions action is optional. This action is normally the first in the sequence, but the AppSearch Action may be sequenced before the LaunchConditions action. If there are launch conditions that do not apply to all installation modes, the appropriate installation mode property should be used in a conditional expression in the appropriate sequence table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [AdminUISequence](#),
[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress [YesNoType](#) If yes, this action will not occur.

See Also

[Wix Schema](#), [Condition](#)

Version 2.0.4820.0

ListBox Element

Description

Set of items for a particular ListBox control tied to an install Property

Windows Installer references

[Control Table](#), [Dialog Table](#), [ListView Table](#)

Parents

[Control](#), [UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ListItem](#) (min: 0, max: unbounded): entry for ListBox table

Attributes

Name	Type	Description	Required
Property	String	Property tied to this group	Yes

See Also

[Wix Schema](#)

Listltem Element

Description

Text and value associated with Property with Control set to ListBox, ListView, ComboBox

Windows Installer references

[Control Table](#), [ComboBox Table](#), [Dialog Table](#), [ListBox Table](#), [ListView Table](#)

Parents

[ComboBox](#), [ListBox](#), [ListView](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Text](#) (min: 0, max: 1): Alternative to Text attribute when CDATA is needed to escape XML delimiters.

Attributes

Name	Type	Description	Required
Icon	String	Only valid in ListView Properties	
Text	String	Defaults to Listltem's value	
Value	String	Value assigned to the associated control Property.	Yes

See Also

[Wix Schema](#)

ListView Element

Description

Set of items for a particular ListView control tied to an install Property

Windows Installer references

[ListView Table](#), [Control Table](#), [Dialog Table](#)

Parents

[Control](#), [UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ListItem](#) (min: 0, max: unbounded): entry for ListView table

Attributes

Name	Type	Description	Required
Property	String	Property tied to this group	Yes

See Also

[Wix Schema](#)

Media Element

Description

Media element describes a disk that makes up the source media for the installation.

Windows Installer references

[Media Table](#)

Parents

[Fragment](#), [Include](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [DigitalSignature](#) (min: 0, max: 1)
2. [PatchPackage](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Integer	Disk identifier for Media table. This number must be equal to or greater than 1.	Yes
Cabinet	String	The name of the cabinet if some or all of the files stored on the media are compressed into a cabinet file. If no cabinets are used, this attribute must be blank.	
CompressionLevel	Enumeration	Indicates the compression level for the Media's cabinet. This	

attribute can only be used in conjunction with the Cabinet attribute. The default is 'mszip'. This attribute's value should be one of the following:

high

low

medium

mszip

none

DiskPrompt	String	The disk name, which is usually the visible text printed on the disk. This localizable text is used to prompt the user when this disk needs to be inserted. This value will be used in the "[1]" of the DiskPrompt Property. Using this attribute will require you to define a DiskPrompt Property.
EmbedCab	YesNoType	Instructs the binder to embed the cabinet in the product if 'yes'. This attribute can only be specified in conjunction with the Cabinet attribute.
Layout	String	This attribute specifies the root directory for the uncompressed files that are a part of this Media

element. By default, the src will be the output directory for the final image. The default value ensures the binder generates an installable image. If a relative path is specified in the src attribute, the value will be appended to the image's output directory. If an absolute path is provided, that path will be used without modification. The latter two options are provided to ease the layout of an image onto multiple medias (CDs/DVDs).

src	String	This attribute has been deprecated; please use the Layout attribute instead.
VolumeLabel	String	The label attributed to the volume. This is the volume label returned by the GetVolumeInformation function. If the SourceDir property refers to a removable (floppy or CD-ROM) volume, then this volume label is used to verify that the proper disk is in the drive before attempting to install files. The entry in this column must match the volume

label of the physical
media.

See Also

[Wix Schema](#)

Version 2.0.4820.0

Merge Element

Description

Merge directive to bring in a Merge Module to be redirected to parent Directory

Windows Installer references

None

Parents

[Directory](#), [DirectoryRef](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [ConfigurationData](#) (min: 0, max: unbounded): Data to use as input to a configurable merge module.

Attributes

Name	Type	Description	Required
Id	String	The unique identifier for the Merge element in the source code. Referenced by the MergeRef/@Id.	Yes
DiskId	String	The value of this attribute should correspond to the Id attribute of a Media element authored elsewhere. By creating this connection between the Merge Module and Media element, you set the packaging options to the values specified in the Media element (values such	Yes

as compression level, cab embedding, etc...).

FileCompression	YesNoType		
Language	Integer	Specifies the decimal LCID for the language to merge the Module in as.	Yes
SourceFile	String		
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

See Also

[Wix Schema](#), [MergeRef](#)

Version 2.0.4820.0

MergeRef Element

Description

Merge reference to connect a Merge Module to parent Feature

Windows Installer references

None

Parents

[Feature](#), [FeatureRef](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The unique identifier for the Merge element to be referenced.	Yes
Primary	YesNoType	Specifies whether the feature containing this MergeRef is the primary feature for advertising the merge module's components.	

See Also

[Wix Schema](#), [Merge](#)

MigrateFeatureStates Element

Description

Used for upgrading or installing over an existing application. Reads feature states from existing application and sets these feature states for the pending installation. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

MIME Element

Description

MIME content-type for an Extension

Windows Installer references

[MIME Table](#)

Parents

[Extension](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Class	Uuid	Class ID for the COM server that is to be associated with the MIME content.	
ContentType	String	This is the identifier for the MIME content. It is commonly written in the form of type/format.	Yes
Default	YesNoType	If 'yes', become the content type for the parent Extension. The default value is 'no'.	

See Also

[Wix Schema](#)

MimeMap Element

Description

MimeMap definition for IIS resources.

Windows Installer references

None

Parents

[WebVirtualDir](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Id for the MimeMap.	Yes
Extension	String	Extension covered by the MimeMap. Must begin with a dot.	Yes
Type	String	Mime-type covered by the MimeMap.	Yes

See Also

[Wix Schema](#)

Module Element

Description

The Module element is analogous to the main function in a C program. When linking, only one Module section can be given to the linker to produce a successful result. Using this element creates an msm file.

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Package](#) (min: 1, max: 1)
2. Choice of elements (min: 0, max: unbounded)
 - [AppId](#) (min: 0, max: unbounded)
 - [Binary](#) (min: 0, max: unbounded)
 - [ComponentGroupRef](#) (min: 0, max: unbounded)
 - [ComponentRef](#) (min: 0, max: unbounded)
 - [Configuration](#) (min: 0, max: unbounded)
 - [CustomAction](#) (min: 0, max: unbounded)
 - [CustomActionRef](#) (min: 0, max: unbounded)
 - [CustomTable](#) (min: 0, max: unbounded)
 - [Dependency](#) (min: 0, max: unbounded)
 - [Directory](#) (min: 0, max: unbounded)
 - [DirectoryRef](#) (min: 0, max: unbounded)
 - [EnsureTable](#) (min: 0, max: unbounded)
 - [Exclusion](#) (min: 0, max: unbounded)

- [FragmentRef](#) (min: 0, max: unbounded)
- [Group](#) (min: 0, max: unbounded)
- [Icon](#) (min: 0, max: unbounded)
- [IgnoreModularization](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [SqlDatabase](#) (min: 0, max: unbounded)
- [Substitution](#) (min: 0, max: unbounded)
- [UI](#) (min: 0, max: unbounded)
- [UIRef](#) (min: 0, max: unbounded)
- [User](#) (min: 0, max: unbounded)
- [WebApplication](#) (min: 0, max: unbounded)
- [WebAppPool](#) (min: 0, max: unbounded)
- [WebDirProperties](#) (min: 0, max: unbounded)
- [WebLog](#) (min: 0, max: unbounded)
- [WebSite](#) (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 1. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- Any Element namespace='###other' processContents='Lax'

Attributes

Name	Type	Description	Required
Id	String	The name of the merge module (not the file name).	Yes
Codepage	Integer	The codepage of the merge module.	
Guid	Uuid	The product code GUID of the merge module.	Yes
Language	LocalizableInteger	The decimal language ID	Yes

		(LCID) of the merge module.	
Version	String	The product version string of the merge module.	Yes

See Also

[Wix Schema](#)

Version 2.0.4820.0

MoveFiles Element

Description

Locates existing files on the system and moves or copies those files to a new location. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

MsiPublishAssemblies Element

Description

Manages the advertisement of CLR and Win32 assemblies. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

MsiUnpublishAssemblies Element

Description

Manages the unadvertisement of CLR and Win32 assemblies that are being removed. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

ODBCDataSource Element

Description

ODBCDataSource for a Component

Windows Installer references

[ODBCDataSource Table](#)

Parents

[Component](#), [Include](#), [ODBCDriver](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Property](#) (min: 0, max: unbounded): Translates into ODBCSourceAttributes

Attributes

Name	Type	Description	Required
Id	String	Identifier of the data source.	Yes
DriverName	String	Required if not found as child of ODBCDriver element	
KeyPath	YesNoType	Set 'yes' to force this file to be key path for parent Component	
Name	String	Name for the data source.	Yes
Registration	Enumeration	Scope for which the data source should be registered. This attribute's value should be one of the following: <i>machine</i> <i>user</i>	Yes

See Also

Wix Schema

Version 2.0.4820.0

ODBCDriver Element

Description

ODBCDriver for a Component

Windows Installer references

[ODBCDriver Table](#)

Parents

[Component](#), [File](#), [Include](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Property](#) (min: 0, max: unbounded): Translates into ODBCSourceAttributes
2. [ODBCDataSource](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the driver.	Yes
File	String	Required if not found as child of File element	
Name	String	Name for the driver.	Yes
SetupFile	String	Required if not found as child of File element or different from File attribute above	

See Also

[Wix Schema](#)

ODBCTranslator Element

Description

ODBCTranslator for a Component

Windows Installer references

[ODBCTranslator Table](#)

Parents

[Component](#), [File](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the translator.	Yes
File	String	Required if not found as child of File element	
Name	String	Name for the translator.	Yes
SetupFile	String	Required if not found as child of File element or different from File attribute above	

See Also

[Wix Schema](#)

Package Element

Description

Properties about the package to be placed in the Summary Information Stream. These are visible from COM through the IStream interface, and these properties can be seen on the package in Explorer.

Windows Installer references

None

Parents

[Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Autogenuuid	Package code GUID for SKU.	Yes
AdminImage	YesNoType	Set to 'yes' if the source is an admin image.	
Comments	String	Optional comments for browsing.	
Compressed	YesNoType	Set to 'yes' to have compressed files in the source.	
Description	String	The product full name or description.	

InstallerVersion	Integer	The minimum installer version (major*100 + minor).
InstallPrivileges	Enumeration	<p>Use this attribute to specify the privileges required to install the package on Windows Vista and above. This attribute's value should be one of the following:</p> <p><i>limited</i> Set this value to declare that the package does not require elevated privileges to install.</p> <p><i>elevated</i> Set this value to declare that the package requires elevated privileges to install. This is the default value.</p>
Keywords	String	Optional keywords for

		browsing.
Languages	String	The list of language IDs (LCIDs) supported in the package.
Manufacturer	String	The vendor releasing the package.
Platforms	String	The list of platforms supported in the package.
ReadOnly	YesNoDefaultType	The value of this attribute conveys whether the package should be opened as read-only. A database editing tool should not modify a read-only enforced database and should issue a warning at attempts to modify a read-only recommended database.
ShortNames	YesNoType	Set to 'yes' to have short filenames in the source.
SummaryCodepage	LocalizableInteger	The codepage for summary info

strings only. The language neutral codepage, zero, is not a valid value.

See Also

[Wix Schema](#)

Version 2.0.4820.0

Patch Element

Description

Patch information for parent File element

Windows Installer references

[Patch Table](#)

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Header	String	stream in Binary table	Yes
PatchSize	Integer	may be defaulted if build tools supply actual size	
Sequence	Integer	may be defaulted if not in cabinet if build tools supply sequence	
Vital	YesNoType		

See Also

[Wix Schema](#)

PatchCertificates Element

Description

Identifies the possible signer certificates used to digitally sign patches.

Windows Installer references

[MsiPatchCertificate Table](#)

Parents

[Fragment](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 1, max: unbounded)

- [DigitalCertificate](#) (min: 1, max: unbounded)

Attributes

None

See Also

[Wix Schema](#)

PatchCreation Element

Description

The PatchCreation element is analogous to the main function in a C program. When linking, only one PatchCreation section can be given to the linker to produce a successful result. Using this element creates a pcg file.

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [PatchInformation](#) (min: 1, max: 1)
2. [PatchMetadata](#) (min: 0, max: 1)
3. [Family](#) (min: 1, max: unbounded)
4. Choice of elements (min: 0, max: unbounded)
 - [PatchProperty](#) (min: 0, max: unbounded)
 - [PatchSequence](#) (min: 0, max: unbounded)
 - [ReplacePatch](#) (min: 0, max: unbounded)
 - [TargetProductCode](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Uuid	Guid for this patch.	Yes
AllowMajorVersionMismatches	YesNoType	True if ProductVersion property may	

		differ by a major version.
AllowProductCodeMismatches	YesNoType	ProductCode property may differ between UpgradedImages table and TargetImages table.
CleanWorkingFolder	YesNoType	Whether patchwiz should clean the temp folder when finished.
Codepage	Integer	The codepage for the resulting PCP.
OutputPath	String	Output patch for patchwiz.
SourceList	String	Used to locate the .msp file for the patch if the cached copy is unavailable.
SymbolFlags	Int	Symbol flags.
WholeFilesOnly	YesNoType	Changing files should be included in their entirety.

See Also

[Wix Schema](#)

PatchFiles Element

Description

Queries the Patch table to determine which patches are to be applied. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

PatchInformation Element

Description

Properties about the patch to be placed in the Summary Information Stream. These are visible from COM through the IStream interface, and these properties can be seen on the package in Explorer.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
AdminImage	YesNoType	Source is an admin image	
Comments	String	Optional comments for browsing	
Compressed	YesNoType	Compressed files on source	
Description	String	Product full name or description	
Keywords	String	Optional keywords for browsing	
Languages	String	List of language IDs supported in package	

Manufacturer	String	Vendor releasing the package
Platforms	String	List of platforms supported in package
ReadOnly	YesNoDefaultType	The value of this attribute conveys whether the package should be opened as read-only. A database editing tool should not modify a read-only enforced database and should issue a warning at attempts to modify a read-only recommended database.
ShortNames	YesNoType	Short filenames on source
SummaryCodepage	LocalizableInteger	The codepage for summary info strings only. The language neutral codepage, zero, is not a valid value.

See Also

[Wix Schema](#)

PatchMetadata Element

Description

Properties about the patch to be placed in the PatchMetadata table.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

Choice of elements (min: 1, max: 1)

- [CustomProperty](#) (min: 0, max: 1): A custom property that extends the standard set.

Attributes

Name	Type	Description	Required
AllowRemoval	YesNoType	Whether this is an uninstallable patch.	Yes
Classification	Enumeration	Category of updates. This attribute's value should be one of the following: <i>Critical Update</i> <i>Hotfix</i> <i>Security Rollup</i> <i>Service Pack</i> <i>Update</i> <i>Update Rollup</i>	Yes

CreationTimeUTC	String	Creation time of the .msp file in the form mm:dd:yy:HH:MM (month: day : year : hour : minute).	
Description	String	Description of the patch.	Yes
DisplayName	String	A title for the patch that is suitable for public display. In Add/Remove Programs from XP SP2 on.	Yes
ManufacturerName	String	Name of the manufacturer.	Yes
MinorUpdateTargetRTM	String	Indicates that the patch targets the RTM version of the product or the most recent major upgrade patch. Author this optional property in minor update patches that contain sequencing information to indicate that the patch removes all patches up to the RTM version of the product, or up to the most recent major upgrade patch. This property is available beginning with Windows Installer	

3.1.

MoreInfoURL	String	A URL that provides information specific to this patch. In Add/Remove Programs from XP SP2 on.
OptimizedInstallMode	YesNoType	If this attribute is set to 'yes' in all the patches to be applied in a transaction, the application of the patch is optimized if possible. Available beginning with Windows Installer 3.1.
TargetProductName	String	Name of the application or target product suite.

See Also
[Wix Schema](#)

PatchPackage Element

Description

PatchPackage found on parent Media element

Windows Installer references

None

Parents

[Media](#)

Inner Text (uuid)

Element value is PatchId GUID.

Children

None

Attributes

None

See Also

[Wix Schema](#)

Version 2.0.4820.0

PatchProperty Element

Description

A property for this patch database.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Name	String	Name of the patch creation property.	Yes
Value	String	Value of the patch creation property.	Yes

See Also

[Wix Schema](#)

PatchSequence Element

Description

Sequence information for this patch database. Sequence information is generated automatically in most cases, and rarely needs to be set explicitly.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
PatchFamily	String	Identifier which indicates one of the sequence families to which this patch belongs.	Yes
Sequence	String	Used to populate the sequence column of the MsiPatchSequence table in the final MSP file. Specified in x.x.x.x format. See documentation for Sequence column of MsiPatchSequence table in MSI SDK.	
Supersede	Integer	Non-NULL value indicates that this patch supersedes earlier patches in this family. See documentation for Attributes column of MsiPatchSequence table in MSI	

		SDK.
Target	String	Used to determine the product code filtering for the patch family.

See Also

[Wix Schema](#)

Version 2.0.4820.0

PerfCounter Element

Description

Used to install Perfmon counters.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Name	String		

See Also

[Wix Schema](#)

Permission Element

Description

Sets ACLs on File, Registry, or CreateFolder. When under a Registry element, this cannot be used if the Action attribute's value is remove or removeKeyOnInstall. This element has no Id attribute. The table and key are taken from the parent element.

Windows Installer references

[LockPermissions Table](#)

Parents

[CreateFolder](#), [File](#), [FileShare](#), [Registry](#), [ServiceInstall](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
Append	YesNoType	
ChangePermission	YesNoType	
CreateChild	YesNoType	For a directory, the right to subdirectory. Only valid under a 'CreateFolder' parent.
CreateFile	YesNoType	For a directory, the right to file in the directory. Only valid under a 'CreateFolder' parent.
CreateLink	YesNoType	
CreateSubkeys	YesNoType	
Delete	YesNoType	
DeleteChild	YesNoType	For a directory, the right to delete files in the directory and all the files in the directory.

		contains, including read-c Only valid under a 'Create parent.
Domain	String	
EnumerateSubkeys	YesNoType	
Execute	YesNoType	
Extended	YesNoType	Specifies whether or not t LockPermissions table w Permission element is ne under a Registry, File, or CreateFolder element. If l is set to 'yes' then the Wi SecureObject custom act used to lock down the res instead of the "legacy" LockPermissions table. S 'yes' for this attribute will r you to link your MSI with t wixca.wixlib. By using the SecureObject custom act can apply permissions for more well known user SIE as for user accounts that created as part of the inst
GenericAll	YesNoType	
GenericExecute	YesNoType	
GenericRead	YesNoType	specifying this will fail to g access
GenericWrite	YesNoType	
Notify	YesNoType	
Read	YesNoType	
ReadAttributes	YesNoType	
ReadExtendedAttributes	YesNoType	
ReadPermission	YesNoType	
ServiceChangeConfig	YesNoType	Required to call the ChangeServiceConfig or

		ChangeServiceConfig2 function to change the service configuration. Only valid under a 'ServiceInstall' parent.
ServiceEnumerateDependents	YesNoType	Required to call the EnumDependentServices function to enumerate all the services dependent on the service. Only valid under a 'ServiceInstall' parent.
ServiceInterrogate	YesNoType	Required to call the ControlServiceStatus function to ask the service for its status immediately. Only valid under a 'ServiceInstall' parent.
ServicePauseContinue	YesNoType	Required to call the ControlService function to pause or continue a service. Only valid under a 'ServiceInstall' parent.
ServiceQueryConfig	YesNoType	Required to call the QueryServiceConfig and QueryServiceConfig2 functions to query the service configuration. Only valid under a 'ServiceInstall' parent.
ServiceQueryStatus	YesNoType	Required to call the QueryServiceStatus function to query the service control manager for the status of the service. Only valid under a 'ServiceInstall' parent.
ServiceStart	YesNoType	Required to call the StartService function to start the service. Only valid under a 'ServiceInstall' parent.
ServiceStop	YesNoType	Required to call the ControlService function to stop the service. Only valid under a 'ServiceInstall' parent.

ServiceUserDefinedControl	YesNoType	Required to call the Control function to specify a user-control code. Only valid under a 'ServiceInstall' parent.
Synchronize	YesNoType	
TakeOwnership	YesNoType	
Traverse	YesNoType	For a directory, the right to traverse the directory. By default, it is assigned the BYPASS_TRAVERSE_CONTROL privilege, which ignores the FILE_TRAVERSE access. Only valid under a 'CreateDirectory' parent.
User	String	
Write	YesNoType	
WriteAttributes	YesNoType	
WriteExtendedAttributes	YesNoType	

See Also

[Wix Schema](#)

ProcessComponents Element

Description

Registers and unregisters components, their key paths, and the component clients. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Product Element

Description

The Product element is analogous to the main function in a C program. When linking, only one Product section can be given to the linker to produce a successful result. Using this element creates an msi file.

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Package](#) (min: 1, max: 1)
2. Choice of elements (min: 0, max: unbounded)
 - [AppId](#) (min: 0, max: unbounded)
 - [Binary](#) (min: 0, max: unbounded)
 - [ComplianceCheck](#) (min: 0, max: unbounded)
 - [Condition](#) (min: 0, max: unbounded)
 - [CustomAction](#) (min: 0, max: unbounded)
 - [CustomActionRef](#) (min: 0, max: unbounded)
 - [CustomTable](#) (min: 0, max: unbounded)
 - [Directory](#) (min: 0, max: unbounded)
 - [DirectoryRef](#) (min: 0, max: unbounded)
 - [EnsureTable](#) (min: 0, max: unbounded)
 - [Feature](#) (min: 0, max: unbounded)
 - [FeatureRef](#) (min: 0, max: unbounded)
 - [FragmentRef](#) (min: 0, max: unbounded)

- [Group](#) (min: 0, max: unbounded)
- [Icon](#) (min: 0, max: unbounded)
- [Media](#) (min: 0, max: unbounded)
- [PatchCertificates](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [SqlDatabase](#) (min: 0, max: unbounded)
- [UI](#) (min: 0, max: unbounded)
- [UIRef](#) (min: 0, max: unbounded)
- [Upgrade](#) (min: 0, max: unbounded)
- [User](#) (min: 0, max: unbounded)
- [WebApplication](#) (min: 0, max: unbounded)
- [WebAppPool](#) (min: 0, max: unbounded)
- [WebDirProperties](#) (min: 0, max: unbounded)
- [WebLog](#) (min: 0, max: unbounded)
- [WebSite](#) (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 1. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- [Any Element namespace='###other' processContents='Lax'](#)
- [HelpCollectionRef](#)
- [HelpFilter](#)

Attributes

Name	Type	Description	Required
Id	Autogenuuid	The product code GUID for the product.	Yes
Codepage	Integer	The codepage for the resulting MSI.	
Language	LocalizableInteger	The decimal language	Yes

		ID (LCID) for the product.	
Manufacturer	String	The manufacturer of the product.	Yes
Name	String	The descriptive name of the product.	Yes
UpgradeCode	Uuid	The upgrade code GUID for the product.	
Version	String	The product's version string.	Yes
<i>Any attribute namespace='###other' processContents='lax'</i>			

See Also

[Wix Schema](#)

Version 2.0.4820.0

ProgId Element

Description

ProgId registration for parent Component. If ProgId has an associated Class, it must be a child of that element.

Windows Installer references

[ProgId Table](#), [Class Table](#), [Registry Table](#), [Icon Table](#)

Parents

[Class](#), [Component](#), [Include](#), [ProgId](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ProgId](#) (min: 0, max: 1): Version-independent ProgId must be child element of actual ProgId. Nesting further ProgId elements within the Version-independent ProgId is disallowed.
2. [Extension](#) (min: 0, max: unbounded): extensions that refer to this ProgId

Attributes

Name	Type	Description	Required
Id	String		Yes
Advertise	YesNoType		
Description	String		
Icon	String	reference to Icon element	
IconIndex	Integer		
NoOpen	String	Specifies that the associated ProgId should not be opened by users. The value is presented as a warning to users. An empty	

string is also valid for this attribute. See [the MSDN documentation](#) for more information.

See Also

[Wix Schema](#)

Version 2.0.4820.0

ProgressText Element

Description

None

Windows Installer references

[ActionText Table](#)

Parents

[UI](#)

Inner Text (xs:string)

Element value is progress message text for action

Children

None

Attributes

Name	Type	Description	Required
Action	String		Yes
Template	String	used to format ActionData messages from action processing	

See Also

[Wix Schema](#)

Property Element

Description

Property value for a Product or Module.

Windows Installer references

[Property Table](#)

Parents

[Control](#), [Fragment](#), [Include](#), [Module](#), [ODBCDataSource](#),
[ODBCDriver](#), [Product](#), [UI](#), [Upgrade](#)

Inner Text (xs:string)

This element may have inner text.

Children

Sequence (min: 1, max: 1)

1. [ComplianceDrive](#) (min: 0, max: 1): Starts searches from the CCP_DRIVE.
2. [ComponentSearch](#) (min: 0, max: unbounded)
3. [RegistrySearch](#) (min: 0, max: unbounded)
4. [RegistrySearchRef](#) (min: 0, max: unbounded)
5. [IniFileSearch](#) (min: 0, max: unbounded)
6. [DirectorySearch](#) (min: 0, max: unbounded)
7. [DirectorySearchRef](#) (min: 0, max: unbounded)
8. [FileSearch](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for Property.	Yes
Admin	YesNoType	Denotes that the Property is saved during administrative installation . See the	

		AdminProperties Property for more information.
ComplianceCheck	YesNoType	Adds a row to the CCPSearch table. This attribute is only valid when this Property contains a search element.
Hidden	YesNoType	Denotes that the Property is not logged during installation. See the MsiHiddenProperties Property for more information.
Secure	YesNoType	Denotes that the Property can be passed to the server side when doing a managed installation with elevated privileges. See the SecureCustomProperties Property for more information.
Value	String	Sets a default value for the property. The value will be overwritten if the Property is used for a search.

See Also

[Wix Schema](#), [PropertyRef](#)

PropertyRef Element

Description

Reference to a Property value.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier of Property to reference.	Yes

See Also

[Wix Schema](#), [Property](#)

Version 2.0.4820.0

ProtectFile Element

Description

Specifies a file to be protected.

Windows Installer references

None

Parents

[Family](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [ProtectRange](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
File	String	Foreign key into the File table.	Yes

See Also

[Wix Schema](#)

ProtectRange Element

Description

Specifies part of a file that cannot be overwritten during patching.

Windows Installer references

None

Parents

[ExternalFile](#), [ProtectFile](#), [TargetFile](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Length	Int	Length of the range.	Yes
Offset	Int	Offset of the start of the range.	Yes

See Also

[Wix Schema](#)

Publish Element

Description

None

Windows Installer references

[ControlEvent Table](#)

Parents

[Control](#)

Inner Text (xs:string)

The element value is the optional Condition expression.

Children

None

Attributes

Name	Type	Description	Required
Event	String	Set this attribute's value to one of the standard control events to trigger that event. Either this attribute or the Property attribute must be set, but not both at the same time.	
Property	String	Set this attribute's value to a property name to set that property. Either this attribute or the Event attribute must be set, but not both at the same time.	
Value	String	If the Property attribute is specified, set the value of this attribute to the new value for the property. To set a property to null, do not set this attribute (the ControlEvent Argument column will be set to '{}'). Otherwise, this attribute's value should be the argument for the event specified in	

the Event attribute.

See Also

[Wix Schema](#)

Version 2.0.4820.0

PublishComponents Element

Description

Manages the advertisement of the components from the PublishComponent table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

PublishFeatures Element

Description

Writes each feature's state into the system registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

PublishProduct Element

Description

Manages the advertisement of the product information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RadioButton Element

Description

Text or Icon plus Value that is assigned to the Property of the parent Control (RadioButtonGroup).

Windows Installer references

[RadioButton Table](#), [Control Table](#), [Dialog Table](#)

Parents

[RadioButtonGroup](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Bitmap	String	This attribute defines the bitmap displayed with the radio button. The value of the attribute creates a reference to a Binary element that represents the bitmap. This attribute is mutually exclusive with the Icon and Text attributes.	
Height	LocalizableInteger		Yes
Help	String		
Icon	String	This attribute defines the icon displayed with the radio button. The value of the attribute creates a reference to a Binary element that	

represents the icon. This attribute is mutually exclusive with the Bitmap and Text attributes.

Text	String	Text displayed with the radio button. This attribute is mutually exclusive with the Bitmap and Icon attributes.	
ToolTip	String		
Value	String	Value assigned to the associated control Property when this radio button is selected.	Yes
Width	LocalizableInteger		Yes
X	LocalizableInteger		Yes
Y	LocalizableInteger		Yes

See Also

[Wix Schema](#), [RadioButtonGroup](#)

RadioButtonGroup Element

Description

Set of radio buttons tied to the specified Property

Windows Installer references

[RadioButton Table](#), [Control Table](#), [Dialog Table](#)

Parents

[Control](#), [UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [RadioButton](#) (min: 1, max: unbounded)

Attributes

Name	Type	Description	Required
Property	String	Property tied to this group.	Yes

See Also

[Wix Schema](#)

RecycleTime Element

Description

IIS6 Application Pool Recycle Times on 24 hour clock.

Windows Installer references

None

Parents

[WebAppPool](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Value	String	Pattern: '\d{1,2}:\d{2}'.	Yes

See Also

[Wix Schema](#)

RegisterClassInfo Element

Description

Manages the registration of COM class information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterComPlus Element

Description

Registers COM+ applications. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterExtensionInfo Element

Description

Manages the registration of extension related information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterFonts Element

Description

Registers installed fonts with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterMIMEInfo Element

Description

Registers MIME-related registry information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterProduct Element

Description

Registers the product information with the installer. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterProgIdInfo Element

Description

Manages the registration of OLE ProgId information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[AdvertiseExecuteSequence](#), [InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterTypeLibraries Element

Description

Registers type libraries with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RegisterUser Element

Description

Registers the user information with the installer to identify the user of a product. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Registry Element

Description

This element allows you to add or remove registry keys (depending upon the value of the action attribute). Please note that for removal, there are 4 options: you can remove a particular registry name, an entire registry key when the parent component is installed, an entire registry key when the parent component is uninstalled, or create a key when the parent component is installed, then remove it when the parent component is uninstalled.

Windows Installer references

[Registry Table](#)

Parents

[Component](#), [Include](#), [Registry](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Permission](#) (min: 0, max: unbounded)
- [Registry](#) (min: 0, max: unbounded)
- [RegistryValue](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Action	Enumeration	This is the action that will be taken for this registry key. This attribute's value should be one of the following: <i>append</i> Appends the specified value(s) to a multiString registry key. <i>createKey</i> Creates the key, if absent, when	

the parent component is installed.

createKeyAndRemoveKeyOnUninstall

Creates the key, if absent, when the parent component is installed then remove the key with all its values and subkeys when the parent component is uninstalled.

prepend

Prepends the specified value(s) to a multiString registry key.

remove

Removes a registry name when the parent component is installed.

removeKeyOnInstall

Removes a key with all its values and subkeys when the parent component is installed.

removeKeyOnUninstall

Removes a key with all its values and subkeys when the parent component is uninstalled.

write

Writes a registry value.

Id	String	Primary key used to identify this particular entry. If this attribute is not specified, an identifier will be generated by hashing the parent Component identifier, Root, Key, and Name.
Key	String	The localizable key for the registry value.
KeyPath	YesNoType	Set this attribute to 'yes' to make this

registry key the KeyPath of the parent component. Only one resource (registry, file, etc) can be the KeyPath of a component.

Name	String	The localizable registry value name. If this attribute is not provided the default value for the registry key will be set instead. The Windows Installer allows several special values to be set for this attribute. You should not use them in WiX. Instead use appropriate values in the Action attribute to get the desired behavior.
Root	Enumeration	<p>The predefined root key for the registry value. This attribute's value should be one of the following:</p> <p><i>HKMU</i></p> <p>A per-user installation will make the operation occur under HKEY_CURRENT_USER. A per-machine installation will make the operation occur under HKEY_LOCAL_MACHINE.</p> <p><i>HKCR</i></p> <p>Operation occurs under HKEY_CLASSES_ROOT. When using Windows 2000 or later, the installer writes or removes the value from the HKCU\Software\Classes hive during per-user installations. When using Windows 2000 or later operating systems, the installer writes or removes the value from the HKLM\Software\Classes hive during per-machine installations.</p>

HKCU

Operation occurs under HKEY_CURRENT_USER. It is recommended to set the KeyPath='yes' attribute when setting this value in order to ensure that the installer writes the necessary registry entries when there are multiple users on the same computer.

HKLM

Operation occurs under HKEY_LOCAL_MACHINE.

HKU

Operation occurs under HKEY_USERS.

Type	Enumeration	<p>Set this attribute to the type of the desired registry key. This attribute must be specified whenever the Value attribute or a child RegistryValue element is specified. This attribute should only be set when the value of the Action attribute does not include the word 'remove'. This attribute's value should be one of the following:</p> <p><i>string</i> The value is interpreted and stored as a string (REG_SZ).</p> <p><i>integer</i> The value is interpreted and stored as an integer (REG_DWORD).</p> <p><i>binary</i> The value is interpreted and stored as a hexadecimal value (REG_BINARY).</p>
------	-------------	---

expandable

The value is interpreted and stored as an expandable string (REG_EXPAND_SZ).

multiString

The value is interpreted and stored as a multiple strings (REG_MULTI_SZ). Please note that this value will only result in a multi-string value if there is more than one registry value or the Action attribute's value is 'append' or 'prepend'. Otherwise a string value will be created.

Value	String	Set this attribute to the localizable registry value. This value is formatted. The Windows Installer allows several special values to be set for this attribute. You should not use them in WiX. Instead use appropriate values in the Type attribute to get the desired behavior. This attribute cannot be specified if the Action attribute's value contains the word 'remove'.
-------	--------	---

See Also

[Wix Schema](#)

RegistrySearch Element

Description

Searches for file, directory or registry key and assigns to value of parent Property

Windows Installer references

[RegLocator Table](#), [Signature Table](#)

Parents

[ComplianceCheck](#), [Property](#)

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [DirectorySearch](#) (min: 0, max: 1)
- [DirectorySearchRef](#) (min: 0, max: 1)
- [FileSearch](#) (min: 0, max: 1)
- [FileSearchRef](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String	Signature to be used for the file, directory or registry key being search for.	Yes
Key	String	Key for the registry value.	Yes
Name	String	Registry value name.	
Root	Enumeration	Root key for the registry value. This attribute's value should be one of the following: <i>HKCR</i> <i>HKCU</i>	Yes

HKLM

HKU

Type	Enumeration	<p>The value must be 'file' if the last child is a FileSearch element and must be 'directory' if last child is a DirectorySearch element. This attribute's value should be one of the following:</p> <p><i>directory</i> Sets a directory path from the registry value.</p> <p><i>file</i> Sets a file path from the registry value.</p> <p><i>raw</i> Sets the raw value from the registry value. Please note that this value will contain a prefix as follows: DWORD: Starts with '#' optionally followed by '+' or '-'. REG_BINARY: Starts with '#x' and the installer converts and saves each hexadecimal digit (nibble) as an ASCII character prefixed by '#x'. REG_EXPAND_SZ: Starts with '#%'. REG_MULTI_SZ: Starts with '[~]' and ends with '[~]'. REG_SZ: No prefix, but if the first character of the registry value is '#', the installer escapes the character by prefixing it with another '#'.</p>	Yes
Win64	YesNoType	Instructs the search to look in the	

64-bit registry when the value is 'yes'. Default is 'no' and search looks in the 32-bit registry.

See Also

[Wix Schema](#), [ComponentSearch](#), [IniFileSearch](#)

Version 2.0.4820.0

RegistrySearchRef Element

Description

References an existing RegistrySearch element.

Windows Installer references

None

Parents

[Property](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Specify the Id of the RegistrySearch to reference.	Yes

See Also

[Wix Schema](#), [RegistrySearch](#)

RegistryValue Element

Description

Use several of these elements to specify each registry value in a multiString registry value. This element cannot be used if the Value attribute is specified unless the Type attribute is set to 'multiString'. The values should go in the text area of the RegistryValue element.

Windows Installer references

[Registry Table](#)

Parents

[Registry](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

[Wix Schema](#)

RemoveDuplicateFiles Element

Description

Deletes files installed by the DuplicateFiles action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveEnvironmentStrings Element

Description

Modifies the values of environment variables. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveExistingProducts Element

Description

Goes through the product codes listed in the ActionProperty column of the Upgrade table and removes the products in sequence. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

RemoveFile Element

Description

Remove a file(s) if the parent component is selected for installation or removal. Multiple files can be removed by specifying a wildcard for the value of the Name attribute. By default, the source directory of the file is the directory of the parent component. This can be overridden by specifying the Directory attribute with a value corresponding to the Id of the source directory, or by specifying the Property attribute with a value corresponding to a property that will have a value that resolves to the full path to the source directory.

Windows Installer references

[RemoveFile Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
Id	String	Primary key used to identify this particular entry.
Directory	String	Overrides the directory of the parent component with a specific Directory. This Directory must exist in the installer database at creation time. This attribute cannot be specified in conjunction with the Property attribute.

LongName [WildcardLongFileNameType](#) If the name of the file(s) to be removed need to be longer than 8.3 format, then this attribute should be specified with the long file name (in addition to the Name attribute which is always required for target systems that might not support long file names). All of the files that match the wild card will be removed from the specified directory. The value is a filename that may also contain the wild card characters "?" for any single character or "*" for zero or more occurrences of any character.

Name [WildcardShortFileNameType](#) This value should be set to the localizable name of the file(s) to be removed. All of the files that match the wild card will be removed from the specified directory. The value is a filename that may also contain the wild card characters "?" for any single character or "*" for zero or more occurrences of any character.

On Enumeration This value determines the time at which the file(s) may be removed. This attribute's value should be one of the following:
install
Removes the file only

when the parent component is being installed
(msiInstallStateLocal or msiInstallStateSource).

uninstall

Removes the file only when the parent component is being removed
(msiInstallStateAbsent)

both

Removes the file when the parent component is being installed or removed.

Property	String	Overrides the directory of the parent component with the value of the specified property. The property should have a value that resolves to the full path of the source directory. The property does not have to exist in the installer database at creation time; it could be created at installation time by a custom action, on the command line, etc. This attribute cannot be specified in conjunction with the Directory attribute.
----------	--------	---

See Also

[Wix Schema](#), [CopyFile](#)

RemoveFiles Element

Description

Removes files previously installed by the InstallFiles action. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveFolder Element

Description

Remove an empty folder if the parent component is selected for installation or removal. By default, the folder is the directory of the parent component. This can be overridden by specifying the Directory attribute with a value corresponding to the Id of the directory, or by specifying the Property attribute with a value corresponding to a property that will have a value that resolves to the full path of the folder.

Windows Installer references

[RemoveFile Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary key used to identify this particular entry.	Yes
Directory	String	Overrides the directory of the parent component with a specific Directory. This Directory must exist in the installer database at creation time. This attribute cannot be specified in conjunction with the Property attribute.	
On	Enumeration	This value determines the time at which the folder may be removed.	Yes

This attribute's value should be one of the following:

install

Removes the folder only when the parent component is being installed (msiInstallStateLocal or msiInstallStateSource).

uninstall

Removes the folder only when the parent component is being removed (msiInstallStateAbsent).

both

Removes the folder when the parent component is being installed or removed.

Property	String	Overrides the directory of the parent component with the value of the specified property. The property should have a value that resolves to the full path of the source directory. The property does not have to exist in the installer database at creation time; it could be created at installation time by a custom action, on the command line, etc. This attribute cannot be specified in conjunction with the Directory attribute.
----------	--------	---

See Also

[Wix Schema](#), [CreateFolder](#)

RemoveFolders Element

Description

Removes any folders linked to components set to be removed or run from source. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveIniValues Element

Description

Removes .ini file information specified for removal in the RemoveIniFile table if the component is set to be installed locally or run from source. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveODBC Element

Description

Removes the data sources, translators, and drivers listed for removal during the installation. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveRegistryValues Element

Description

Removes a registry value that has been authored into the registry table if the associated component was installed locally or as run from source, and is now set to be uninstalled. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

RemoveShortcuts Element

Description

Manages the removal of an advertised shortcut whose feature is selected for uninstallation or a nonadvertised shortcut whose component is selected for uninstallation. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

ReplacePatch Element

Description

A patch that is deprecated by this patch.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Uuid	Patch GUID to be unregistered if it exists on the machine targeted by this patch.	Yes

See Also

[Wix Schema](#)

ReserveCost Element

Description

Disk cost to reserve in a folder for running locally and/or from source

Windows Installer references

[ReserveCost Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Directory	String	Defaults to Directory of parent Component.	
RunFromSource	Integer	The number of bytes of disk space to reserve if the component is installed to run from source.	Yes
RunLocal	Integer	The number of bytes of disk space to reserve if the component is installed to run locally.	Yes

See Also

[Wix Schema](#)

ResolveSource Element

Description

Determines the location of the source and sets the SourceDir property if the source has not been resolved yet. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[AdminExecuteSequence](#), [InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

RMCCPSearch Element

Description

Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed. The RMCCPSearch action should be authored into the InstallUISequence table and InstallExecuteSequence table. The installer prevents RMCCPSearch from running in the InstallExecuteSequence sequence if the action has already run in InstallUISequence sequence. The RMCCPSearch action requires the CCP_DRIVE property to be set to the root path on the removable volume that has the installation for any of the qualifying products. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress [YesNoType](#) If yes, this action will not occur.

See Also

[Wix Schema](#), [CCPSearch](#), [ComplianceCheck](#)

Version 2.0.4820.0

Row Element

Description

Row data for a Custom Table

Windows Installer references

None

Parents

[CustomTable](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Data](#) (min: 1, max: unbounded)

Attributes

None

See Also

[Wix Schema](#)

ScheduleReboot Element

Description

Prompts the user to restart the system at the end of installation. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

SelfRegModules Element

Description

Processes all modules listed in the SelfReg table and registers all installed modules with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

SelfUnregModules Element

Description

Unregisters all modules listed in the SelfReg table that are scheduled to be uninstalled. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

ServiceArgument Element

Description

Argument used in ServiceControl parent

Windows Installer references

[ServiceControl Table](#)

Parents

[ServiceControl](#)

See Also

[Wix Schema](#)

Version 2.0.4820.0

ServiceConfig Element

Description

Service configuration information for failure actions.

Windows Installer references

None

Parents

[Component](#), [Include](#), [ServiceInstall](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
FirstFailureActionType	Enumeration	Action to take on the first failure of the service. This attribute's value should be one of the following: <i>none</i> <i>reboot</i> <i>restart</i> <i>runCommand</i>	Yes
ProgramCommandLine	String	If any of the three *ActionType attributes is "runCommand",	

		this specifies the command to run when doing so.	
RebootMessage	String	If any of the three *ActionType attributes is "reboot", this specifies the message to broadcast to server users before doing so.	
ResetPeriodInDays	Integer	Number of days after which to reset the failure count to zero if there are no failures.	
RestartServiceDelayInSeconds	Integer	If any of the three *ActionType attributes is "restart", this specifies the number of seconds to wait before doing so.	
SecondFailureActionType	Enumeration	Action to take on the second failure of the service. This attribute's value should be one	Yes

of the following:
none
reboot
restart
runCommand

ServiceName	String	Required if not under a ServiceInstall element.	
ThirdFailureActionType	Enumeration	Action to take on the third failure of the service. This attribute's value should be one of the following: <i>none</i> <i>reboot</i> <i>restart</i> <i>runCommand</i>	Yes

Remarks

Nesting a ServiceConfig element under a ServiceInstall element will result in the service being installed to be configured.

Nesting a ServiceConfig element under a component element will result in an already installed service to be configured. If the service does not exist prior to the install of the MSI package, the install will fail.

See Also

[Wix Schema](#)

ServiceControl Element

Description

Starts, stops, and removes services for parent Component. This element is used to control the state of a service installed by the MSI or MSM file by using the start, stop and remove attributes. For example, Start='install' Stop='both' Remove='uninstall' would mean: start the service on install, remove the service when the product is uninstalled, and stop the service both on install and uninstall.

Windows Installer references

[ServiceControl Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ServiceArgument](#) (min: 0, max: unbounded): Ordered list of arguments used when modifying services.

Attributes

Name	Type	Description	Required
Id	String		Yes
Name	String	Name of the service.	Yes
Remove	Enumeration	Specifies whether the service should be removed on install, uninstall or both. This attribute's value should be one of the following: <i>install</i> <i>uninstall</i>	

both

Start	Enumeration	Specifies whether the service should be started on install, uninstall or both. This attribute's value should be one of the following: <i>install</i> <i>uninstall</i> <i>both</i>
Stop	Enumeration	Specifies whether the service should be stopped on install, uninstall or both. This attribute's value should be one of the following: <i>install</i> <i>uninstall</i> <i>both</i>
Wait	YesNoType	Specifies whether or not to wait for the service to complete before continuing.

See Also

[Wix Schema](#)

ServiceDependency Element

Description

Service or group of services that must start before the parent service.

Windows Installer references

[ServiceInstall Table](#)

Parents

[ServiceInstall](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The value of this attribute should be one of the following: <ol style="list-style-type: none">1. The name (not the display name) of a previously installed service.2. A foreign key referring to another ServiceInstall/@Id.3. A group of services (in which case the Group attribute should be set to 'yes').	Yes
Group	YesNoType	Set to 'yes' to indicate that the value in the Id attribute is the name of a group of services.	

See Also

Wix Schema

Version 2.0.4820.0

ServiceInstall Element

Description

Adds and removes services for parent Component.

Windows Installer references

[ServiceInstall Table](#)

Parents

[Component](#), [Include](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Permission](#) (min: 0, max: unbounded): Permissions for this service.
2. [ServiceConfig](#) (min: 0, max: 1): Service Config: failure actions for service
3. [ServiceDependency](#) (min: 0, max: unbounded): ordered list of dependencies when installing services

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for this service.	Yes
Account	String	The account under which to start the service. Valid only when ServiceType is ownProcess.	
Arguments	String	Contains any command line arguments or properties required to run the service.	
Description	String	Sets the description of the	

		service.	
DisplayName	String	This column is the localizable string that user interface programs use to identify the service.	
EraseDescription	YesNoType	Determines whether the existing service description will be ignored. If 'yes', the service description will be null, even if the Description attribute is set.	
ErrorControl	Enumeration	Determines what action should be taken on an error. This attribute's value should be one of the following: <i>ignore</i> <i>normal</i> <i>critical</i>	Yes
Interactive	YesNoType	Whether or not the service interacts with the desktop.	
LoadOrderGroup	String	The load ordering group that this service should be a part of.	
Name	String	This column is the string that gives the service name to install.	Yes
Password	String	The password for the account. Valid only when the account has a password.	
Start	Enumeration	Determines when the service should be started. The Windows Installer	Yes

does not support boot or system. This attribute's value should be one of the following:

auto

demand

disabled

boot

system

Type	Enumeration	The Windows Installer does not currently support kernelDriver or systemDriver This attribute's value should be one of the following: <i>ownProcess</i> <i>shareProcess</i> <i>kernelDriver</i> <i>systemDriver</i>	Yes
Vital	YesNoType	The overall install should fail if this service fails to install.	

Remarks

The service executable installed will point to the KeyPath for the Component. Therefore, you must ensure that the correct executable is either the first child File element under this Component or explicitly mark the appropriate File element as KeyPath='yes'.

See Also

[Wix Schema](#)

SetODBCFolders Element

Description

Checks for existing ODBC drivers and sets the target directory for each new driver to the location of an existing driver. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

SFPCatalog Element

Description

Adds a system file protection update catalog file

Windows Installer references

[SFPCatalog Table](#)

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#), [SFPCatalog](#)

Inner Text (xs:string)

Element value can be hex-encoded hash value

Children

Choice of elements (min: 0, max: unbounded)

- [SFPCatalog](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded): Primary Key to File Table.

Attributes

Name	Type	Description	Required
Dependency	String	Used to define dependency outside of the package.	
Name	String	Filename for catalog file when installed.	
SourceFile	String	Path to catalog file in binary.	

See Also

[Wix Schema](#)

SFPFile Element

Description

Provides a many-to-many mapping from the SFPCatalog table to the File table

Windows Installer references

[FileSFPCatalog Table](#)

Parents

[SFPCatalog](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key to File Table.	Yes

See Also

[Wix Schema](#)

Shortcut Element

Description

Shortcut, default target is parent File, CreateFolder, or Component's Directory

Windows Installer references

[Shortcut Table](#)

Parents

[Component](#), [CreateFolder](#), [File](#), [Include](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Icon](#) (min: 0, max: 1)

Attributes

Name	Type	Description
Id	String	Unique identifier for the shortcut. This value will serve as the primary key for the row.
Advertise	YesNoType	Specifies if the shortcut should be advertised or not. Note that advertised shortcuts always point at a particular application, identified by a ProductCode, and should not be shared between applications. Advertised shortcuts

only work for the most recently installed application, and are removed when that application is removed.

Arguments	String	The command-line arguments for the shortcut. Note that the resolution of properties in the Arguments field is limited. A property formatted as [Property] in this field can only be resolved if the property already has the intended value when the component owning the shortcut is installed. For example, for the argument "[#MyDoc.doc]" to resolve to the correct value, the same process must be installing the file MyDoc.doc and the component that owns the shortcut.
Description	String	The localizable description for the shortcut.
DescriptionResourceDll	String	The Formatted string providing the full path to the language neutral file containing the MUI Manifest. Generally authored using [#filekey] form. When

this attribute is specified, the DescriptionResourceId attribute must also be provided.

This attribute is only used on Windows Vista and above. If this attribute is not specified and the install is running on Vista and above, the value in the Name attribute is used. If this attribute is provided and the install is running on Vista and above, the value in the Name attribute is ignored.

DescriptionResourceId Integer

The description name index for the shortcut. This must be a non-negative number. When this attribute is specified, the DescriptionResourceDll attribute must also be populated.

This attribute is only used on Windows Vista and above. If this attribute is not specified and the install is running on Vista and above, the value in the Name attribute is used. If this attribute is

populated and the install is running on Vista and above, the value in the Name attribute is ignored.

Directory	String	Identifier reference to Directory element where shortcut is to be created.
DisplayResourceDll	String	<p>The Formatted string providing the full path to the language neutral file containing the MUI Manifest. Generally authored using [#filekey] form. When this attribute is specified, the DisplayResourceId attribute must also be provided.</p> <p>This attribute is only used on Windows Vista and above. If this attribute is not populated and the install is running on Vista and above, the value in the Name attribute is used. If this attribute is populated and the install is running on Vista and above, the value in the Name attribute is ignored.</p>

DisplayResourceId	Integer	<p>The display name index for the shortcut. This must be a non-negative number. When this attribute is specified, the DisplayResourceDll attribute must also be provided.</p> <p>This attribute is only used on Windows Vista and above. If this attribute is not specified and the install is running on Vista and above, the value in the Name attribute is used. If this attribute is specified and the install is running on Vista and above, the value in the Name attribute is ignored.</p>
Hotkey	Integer	<p>The hotkey for the shortcut. The low-order byte contains the virtual-key code for the key, and the high-order byte contains modifier flags. This must be a non-negative number. Authors of installation packages are generally recommend not to set this option, because this can add duplicate hotkeys to a users</p>

desktop. In addition, the practice of assigning hotkeys to shortcuts can be problematic for users using hotkeys for accessibility.

Icon	String	Identifier reference to Icon element. The Icon identifier should have the same extension as the file that it points at. For example, a shortcut to an executable (e.g. "my.exe") should reference an Icon with identifier like "MyIcon.exe"
IconIndex	Integer	Identifier reference to Icon element.
LongName	LongFileNameType	Localizable long name for shortcut if a name longer than 8.3 format is desired.
Name	ShortFileNameType	Localizable short name for the shortcut. Must be an 8.3 file name.
Show	Enumeration	This attribute's value should be one of the following: <i>normal</i> <i>minimized</i> <i>maximized</i>
Target	String	The target for a non-Advertised shortcut.

This attribute is not valid for Advertised shortcuts. The value will be defaulted to the parent File when nested under a File element. If you specify this value then use a formatted file identifier, for example: [!TargetFileId].

WorkingDirectory	String	Directory identifier (or Property identifier that resolves to a directory) that resolves to the path of the working directory for the shortcut.
------------------	--------	---

See Also
[Wix Schema](#)

Show Element

Description

None

Windows Installer references

None

Parents

[AdminUISequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
After	String		
Before	String		
Dialog	String		Yes
OnExit	Enumeration	mutually exclusive with Before, After, and Sequence attributes This attribute's value should be one of the following: <i>success</i> <i>cancel</i> <i>error</i> <i>suspend</i>	
Sequence	Integer		

See Also

[Wix Schema](#)

Version 2.0.4820.0

SqlDatabase Element

Description

SQL Database

Windows Installer references

None

Parents

[Component](#), [Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [SqlFileSpec](#) (min: 0, max: unbounded)
- [SqlLogFileSpec](#) (min: 0, max: unbounded)
- [SqlScript](#) (min: 0, max: unbounded)
- [SqlString](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
ConfirmOverwrite	YesNoType		
ContinueOnError	YesNoType		
CreateOnInstall	YesNoType		
CreateOnReinstall	YesNoType	Specifies whether to create the database when the associated component is reinstalled. Setting CreateOnInstall to yes does not imply CreateOnReinstall is set to yes. CreateOnReinstall	

must be set in addition to CreateOnInstall for it to be created during both install and reinstall.

CreateOnUninstall	YesNoType		
Database	String	The name of the database. Yes If the name does not follow the SQL server "Rules for Regular Identifiers" (see MSDN) it must be surrounded by quotes or square brackets. Since this value can be formatted text, this means that if you choose to use square brackets you must use the MSI method for escaping square brackets, for example: []blah[].	
DropOnInstall	YesNoType		
DropOnReinstall	YesNoType	Specifies whether to drop the database when the associated component is reinstalled. Setting DropOnInstall to yes does not imply DropOnReinstall is set to yes. DropOnReinstall must be set in addition to DropOnInstall for it to be dropped during both install and reinstall.	
DropOnUninstall	YesNoType		
Instance	String		
Server	String		Yes
User	String		

Remarks

Nesting SqlDatabase under a Component element will result in a SqlDatabase being installed to the machine as the package is installed.

Nesting SqlDatabase under Product, Fragment, or Module results in a database "locator" record being created in the SqlDatabase table. This means that the database itself is neither installed nor uninstalled by the MSI package. It does make the database available for referencing from a SqlString or SqlScript record. This allows MSI to install SqlScripts or SqlStrings to already existing databases on the machine. The install will fail if the database does not exist in these cases.

The User attribute references credentials specified in a User element. If a user is not specified then Windows Authentication will be used by default using the credentials of the user performing the install to execute sql strings, etc.

See Also

[Wix Schema](#), [User](#)

SqlFileSpec Element

Description

File specification for a Sql database.

Windows Installer references

None

Parents

[SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	ID of the file specification.	Yes
Filename	String	Specifies the operating-system file name for the database file.	Yes
GrowthSize	String	Specifies the growth increment of the database file. The GrowthSize setting for a file cannot exceed the MaxSize setting.	
MaxSize	String	Specifies the maximum size to which the database file can grow.	
Name	String	Specifies the logical name for the database file.	
Size	String	Specifies the size of the database file. When a Size is not supplied for a database file, SQL Server uses the size of the primary file in the model database.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

SqlLogFileSpec Element

Description

File specification for a Sql database.

Windows Installer references

None

Parents

[SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Filename	String	Specifies the operating-system file name for the log file.	
GrowthSize	String	Specifies the growth increment of the log file. The GrowthSize setting for a file cannot exceed the MaxSize setting.	
Id	String	ID of the log file specification.	
MaxSize	String	Specifies the maximum size to which the log file can grow.	
Name	String	Specifies the logical name for the log file.	
Size	String	Specifies the size of the log file. When a Size parameter is not specified for a log file, SQL Server makes the file 1 MB.	

See Also

Wix Schema

Version 2.0.4820.0

SqlScript Element

Description

SQL Script

Windows Installer references

None

Parents

[Component](#), [Include](#), [SqlDatabase](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Binary](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String		Yes
BinaryKey	String	Reference to Binary stream that contains the SQL script to execute. Only valid if no Binary child element.	
ContinueOnError	YesNoType	Continue executing scripts even if this one fails.	
ExecuteOnInstall	YesNoType	Specifies to execute the script when the associated component is installed.	
ExecuteOnReinstall	YesNoType	Specifies whether to execute the script when the associated	

component is reinstalled. Setting ExecuteOnInstall to yes does **not** imply ExecuteOnReinstall is set to yes. ExecuteOnReinstall must be set in addition to ExecuteOnInstall for it to be executed during both install and reinstall.

ExecuteOnReInstall	YesNoType	This attribute has been deprecated; please use the ExecuteOnReinstall attribute instead.
ExecuteOnUninstall	YesNoType	Specifies to execute the script when the associated component is uninstalled.
RollbackOnInstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to install the associated component.
RollbackOnReinstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to reinstall the associated component.
RollbackOnUninstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to uninstall the associated component.
Sequence	Integer	Specifies the order to run the SQL Scripts. It is recommended that rollback scripts be

scheduled before their complementary execution script. This order is also relative across the SqlString element.

SqlDb	String	Required when not child of SqlDatabase.
User	String	

See Also

[Wix Schema](#)

SqlString Element

Description

SQL String

Windows Installer references

None

Parents

[Component](#), [Include](#), [SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
ContinueOnError	YesNoType	Continue executing strings even if this one fails.	
ExecuteOnInstall	YesNoType	Specifies to execute the string when the associated component is installed.	
ExecuteOnReinstall	YesNoType	Specifies whether to execute the string when the associated component is reinstalled. Setting ExecuteOnInstall to yes does not imply ExecuteOnReinstall is set to yes. ExecuteOnReinstall	

must be set in addition to ExecuteOnInstall for it to be executed during both install and reinstall.

ExecuteOnReInstall	YesNoType	This attribute has been deprecated; please use the ExecuteOnReinstall attribute instead.
ExecuteOnUninstall	YesNoType	Specifies to execute the string when the associated component is uninstalled.
RollbackOnInstall	YesNoType	Specifies whether to execute the string on rollback if an attempt is made to install the associated component.
RollbackOnReinstall	YesNoType	Specifies whether to execute the string on rollback if an attempt is made to reinstall the associated component.
RollbackOnUninstall	YesNoType	Specifies whether to execute the string on rollback if an attempt is made to uninstall the associated component.
Sequence	Integer	Specifies the order to run the SQL Strings. It is recommended that rollback strings be scheduled before their complementary execution string. This order is also relative across the SqlScript element.

SQL	String	Yes
SqlDb	String	
User	String	

See Also

[Wix Schema](#)

Version 2.0.4820.0

StartServices Element

Description

Starts system services. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

StopServices Element

Description

Stops system services. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Subscribe Element

Description

Sets attributes for events in the EventMapping table

Windows Installer references

[EventMapping Table](#)

Parents

[Control](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Attribute	String	if not present can only handle enable, disable, hide, unhide events	
Event	String	must be one of the standard control events'	

See Also

[Wix Schema](#)

Substitution Element

Description

Specifies the configurable fields of a module database and provides a template for the configuration of each field.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Column	String	Specifies the target column in the row named in the Row column.	Yes
Row	String	Specifies the primary keys of the target row in the table named in the Table column. If multiple keys, separated by semicolons.	Yes
Table	String	Specifies the name of the table being modified in the module database.	Yes
Value	String	Provides a formatting template for the data being substituted into the target field specified by Table, Row, and Column.	

See Also

[Wix Schema](#)

SymbolPath Element

Description

A path to symbols.

Windows Installer references

None

Parents

[ExternalFile](#), [TargetFile](#), [TargetImage](#), [UpgradeFile](#), [UpgradeImage](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Path	String	The path.	Yes

See Also

[Wix Schema](#)

TargetFile Element

Description

Information about specific files in a target image.

Windows Installer references

None

Parents

[TargetImage](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [SymbolPath](#) (min: 0, max: 1)
2. Choice of elements (min: 0, max: unbounded)
 - [IgnoreRange](#) (min: 0, max: unbounded)
 - [ProtectRange](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Foreign key into the File table.	Yes

See Also

[Wix Schema](#)

TargetImage Element

Description

Contains information about the target images of the product.

Windows Installer references

None

Parents

[UpgradeImage](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [SymbolPath](#) (min: 0, max: unbounded)
- [TargetFile](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the target image.	Yes
IgnoreMissingFiles	YesNoType	Files missing from the target image are ignored by the installer.	
Order	Int	Relative order of the target image.	Yes
SourceFile	String	Full path to the location of the msi file for the target image.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

Validation

String

Product checking to avoid applying irrelevant transforms.

See Also

[Wix Schema](#)

Version 2.0.4820.0

TargetProductCode Element

Description

A product code for a product that may receive this patch (or '*' for all products).

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The product code for a product that can receive this patch (or '*' for all products).	Yes

See Also

[Wix Schema](#)

Text Element

Description

Alternative to Text attributes when CDATA is needed to escape XML delimiters.

Windows Installer references

None

Parents

[Control](#), [ListItem](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
SourceFile	String	Instructs the text to be imported from a file instead of the element value during the binding process.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

See Also

[Wix Schema](#)

TextStyle Element

Description

None

Windows Installer references

[TextStyle Table](#)

Parents

[UI](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Blue	Integer	0 to 255	
Bold	YesNoType		
FaceName	String		Yes
Green	Integer	0 to 255	
Italic	YesNoType		
Red	Integer	0 to 255	
Size	Integer		Yes
Strike	YesNoType		
Underline	YesNoType		

See Also

[Wix Schema](#)

TypeLib Element

Description

Register a type library (TypeLib). Please note that in order to properly use this non-advertised, you will need use this element with Advertise='no' and also author the appropriate child Interface elements by extracting them from the type library itself.

Windows Installer references

[TypeLib Table](#), [Registry Table](#)

Parents

[Component](#), [File](#), [Include](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Appld](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [Interface](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Uuid	The GUID that identifies the type library.	Yes
Advertise	YesNoType	Value of 'yes' will create a row in the TypeLib table. Value of 'no' will create rows in the Registry table.	
Control	YesNoType	Value of 'yes' means the type library describes controls, and should not be displayed in type browsers intended for nonvisual objects. This	

		attribute can only be set if Advertise='no'.	
Cost	Int	The cost associated with the registration of the type library in bytes. This attribute cannot be set if Advertise='no'.	
Description	String	The localizable description of the type library.	
HasDiskImage	YesNoType	Value of 'yes' means the type library exists in a persisted form on disk. This attribute can only be set if Advertise='no'.	
HelpDirectory	String	The identifier of the Directory element for the help directory.	
Hidden	YesNoType	Value of 'yes' means the type library should not be displayed to users, although its use is not restricted. Should be used by controls. Hosts should create a new type library that wraps the control with extended properties. This attribute can only be set if Advertise='no'.	
Language	Integer	The language of the type library. This must be a non-negative integer.	Yes
MajorVersion	String	The major version of the type library. The value should be an integer from 0 - 255.	
MinorVersion	String	The minor version of the type library. The value should be an integer from 0 - 255.	
ResourceId	Integer	The resource id of a typelib. The value is appended to the end of the typelib path in the	

registry.

Restricted	YesNoType	Value of 'yes' means the type library is restricted, and should not be displayed to users. This attribute can only be set if Advertise='no'.
------------	---------------------------	--

See Also

[Wix Schema](#)

Version 2.0.4820.0

UI Element

Description

Enclosing element to compartmentalize UI specifications.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [BillboardAction](#) (min: 0, max: unbounded): Billboard table item with child Controls
- [Binary](#) (min: 0, max: unbounded)
- [ComboBox](#) (min: 0, max: unbounded): ComboBox table with ListItem children
- [Dialog](#) (min: 0, max: unbounded): Dialog specification, called from Sequence
- [DialogRef](#) (min: 0, max: unbounded): Reference to a Dialog specification.
- [Error](#) (min: 0, max: unbounded): Error text associated with install error
- [ListBox](#) (min: 0, max: unbounded): ListBox table with ListItem children
- [ListView](#) (min: 0, max: unbounded): ListView table with ListItem children
- [ProgressText](#) (min: 0, max: unbounded): ActionText entry associated with an action
- [Property](#) (min: 0, max: unbounded)
- [RadioButtonGroup](#) (min: 0, max: unbounded): RadioButton table with

RadioButton children

- [TextStyle](#) (min: 0, max: unbounded): TextStyle entry for use in control text
- [UIText](#) (min: 0, max: unbounded): values for UIText property, not installer Property
- Sequence (min: 1, max: 1)
 1. [AdminUISequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String		

See Also

[Wix Schema](#), [UIRef](#)

Version 2.0.4820.0

UIRef Element

Description

Reference to a UI element. This will force the entire referenced Fragment's contents to be included in the installer database.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes

See Also

[Wix Schema](#), [UI](#)

UIText Element

Description

Text associated with certain controls

Windows Installer references

[UIText Table](#)

Parents

[UI](#)

Inner Text (xs:string)

Element value is text, may use CDATA if needed to escape XML delimiters

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes

See Also

[Wix Schema](#)

UnpublishComponents Element

Description

Manages the unadvertisement of components listed in the PublishComponent table. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnpublishFeatures Element

Description

Removes selection-state and feature-component mapping information from the registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterClassInfo Element

Description

Manages the removal of COM class information from the system registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterComPlus Element

Description

Removes COM+ applications from the registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterExtensionInfo Element

Description

Manages the removal of extension-related information from the system registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterFonts Element

Description

Removes registration information about installed fonts from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterMIMEInfo Element

Description

Unregisters MIME-related registry information from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterProgIdInfo Element

Description

Manages the unregistration of OLE ProgId information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

UnregisterTypeLibraries Element

Description

Unregisters type libraries from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Upgrade Element

Description

Upgrade info for a particular UpgradeCode

Windows Installer references

[Upgrade Table](#)

Parents

[Fragment](#), [Include](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [Property](#) (min: 0, max: unbounded): Property table entry for the ActionProperty column associated with this Upgrade row
- [UpgradeVersion](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Uuid	This value specifies the upgrade code for the products that are to be detected by the FindRelatedProducts action.	Yes

See Also

[Wix Schema](#)

UpgradeFile Element

Description

Specifies files to either ignore or to specify optional data about a file.

Windows Installer references

None

Parents

[UpgradedImage](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [SymbolPath](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
AllowIgnoreOnError	YesNoType	Specifies whether patching this file is vital.	
File	String	Foreign key into the File table.	Yes
Ignore	YesNoType	If yes, the file is ignored during patching, and the next two attributes are ignored.	Yes
WholeFile	YesNoType	Whether the whole file should be installed, rather than creating a binary patch.	

See Also

[Wix Schema](#)

UpgradeImage Element

Description

Contains information about the upgraded images of the product.

Windows Installer references

None

Parents

[Family](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [TargetImage](#) (min: 1, max: unbounded)
2. Choice of elements (min: 0, max: unbounded)
 - [SymbolPath](#) (min: 0, max: unbounded)
 - [UpgradeFile](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier to connect target images with upgraded image.	Yes
SourceFile	String	Full path to location of msi file for upgraded image.	
SourcePatch	String	Modified copy of the upgraded installation database that contains additional authoring specific to patching.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	
srcPatch	String	This attribute has been	

deprecated; please use the
SourcePatch attribute instead.

See Also

[Wix Schema](#)

Version 2.0.4820.0

UpgradeVersion Element

Description

None

Windows Installer references

[Upgrade Table](#)

Parents

[Upgrade](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Require
ExcludeLanguages	YesNoType	Set to "yes" to detect all languages, excluding the languages listed in the Language attribute.	
IgnoreRemoveFailure	YesNoType	Set to "yes" to continue installation upon failure to remove a product or application.	
IncludeMaximum	YesNoType	Set to "yes" to make the range of versions detected include the value specified in Maximum.	
IncludeMinimum	YesNoType	Set to "yes" to make the range of versions detected include the value specified in	

		Minimum. This attribute is "yes" by default.
Language	String	Specifies the set of languages detected by FindRelatedProducts. Enter a list of numeric language identifiers (LANGID) separated by commas (,). Leave this value null to specify all languages. Set ExcludeLanguages to "yes" in order detect all languages, excluding the languages listed in this value.
Maximum	String	Specifies the upper boundary of the range of product versions detected by FindRelatedProducts.
MigrateFeatures	YesNoType	Set to "yes" to migrate feature states from upgraded products by enabling the logic in the MigrateFeatureStates action.
Minimum	String	Specifies the lower bound on the range of product versions to be detected by FindRelatedProducts.
OnlyDetect	YesNoType	Set to "yes" to detect products and applications but do not uninstall.
Property	String	When the

FindRelatedProducts action detects a related product installed on the system, it appends the product code to the property specified in this field. The property specified in this field must be a public property and the package author must add the property to the SecureCustomProperties Property. Each UpgradeVersion must have a unique Property value. After FindRelatedProducts the value of this property is a list product codes, separated by semicolons (;), detected on the system.

RemoveFeatures

String

The installer sets the REMOVE property to features specified in this column. The features to be removed can be determined at run time. The Formatted string entered in this field must evaluate to a comma-delimited list of feature names. For example: [Feature1],[Feature2],[Feature3]. No features are removed if the field contains formatted text

that evaluates to an empty string. The installer sets REMOVE=ALL only if the Remove field is empty.

Any attribute namespace='##other' processContents='lax'

See Also

[Wix Schema](#)

Version 2.0.4820.0

User Element

Description

User for all kinds of things. When it is not nested under a component it is included in the MSI so it can be referenced by other elements such as the User attribute in the AppPool element. When it is nested under a Component element, the User will be created on install and can also be used for reference.

Windows Installer references

None

Parents

[Component](#), [Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [GroupRef](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
CanNotChangePassword	YesNoType		
CreateUser	YesNoType	Indicates whether or not to create the user. User creation can be skipped if all that is desired is to join a user to groups.	
Disabled	YesNoType		
Domain	String		
FailIfExists	YesNoType	Indicates if the	

		install should fail if the user already exists.	
Name	String		Yes
Password	String	Usually a Property that is passed in on the command-line to keep it more secure.	
PasswordExpired	YesNoType	Indicates whether the user must change their password on their first login.	
PasswordNeverExpires	YesNoType		
RemoveOnUninstall	YesNoType	Indicates whether the user account should be left behind on uninstall.	
UpdateIfExists	YesNoType	Indicates if the user account properties should be updated if the user already exists.	

See Also

[Wix Schema](#), [Group](#), [GroupRef](#)

ValidateProductID Element

Description

Sets the ProductID property to the full product identifier. This action must be sequenced before the user interface wizard in the InstallUISequence table and before the RegisterUser action in the InstallExecuteSequence table. If the product identifier has already been validated successfully, the ValidateProductID action does nothing. The ValidateProductID action always returns a success, whether or not the product identifier is valid, so that the product identifier can be entered on the command line the first time the product is run. The product identifier can be validated without having the user reenter this information by setting the PIDKEY property on the command line or by using a transform. The display of the dialog box requesting the user to enter the product identifier can then be made conditional upon the presence of the ProductID property, which is set when the PIDKEY property is validated. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#), [InstallUISequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

Verb Element

Description

Verb definition for an Extension. When advertised, this element creates a row in the [Verb table](#). When not advertised, this element creates the appropriate rows in [Registry table](#).

Windows Installer references

[Verb Table](#), [Registry Table](#)

Parents

[Extension](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The verb for the command.	Yes
Argument	String	Value for the command arguments. Note that the resolution of properties in the Argument field is limited. A property formatted as [Property] in this field can only be resolved if the property already has the intended value when the component owning the verb is installed. For example, for the argument "[#MyDoc.doc]" to resolve to the correct value, the same process must be installing the file MyDoc.doc and the component that owns the verb.	
Command	String	The localized text displayed on the	

		context menu.
Sequence	Integer	The sequence of the commands. Only verbs for which the Sequence is specified are used to prepare an ordered list for the default value of the shell key. The Verb with the lowest value in this column becomes the default verb. Used only for Advertised verbs.
Target	String	Target file to be executed for the verb. The value should be a formatted Property to refer to the short path to the file, for example: [!TargetFileId]. Only valid for non-Advertised verbs.

See Also
[Wix Schema](#)

Version 2.0.4820.0

WebAddress Element

Description

WebAddress for WebSite

Windows Installer references

None

Parents

[WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Header	String		
IP	String	For IP address "All Unassigned", do not specify this attribute or specify its value as "*".	
KeyPath	YesNoType		
Port	String		Yes
Secure	YesNoType		

See Also

[Wix Schema](#)

WebApplication Element

Description

Defines properties for a web application. These properties can be used for more than one application defined in a web site, directory, or vroot, by defining this element in a common location and referring to it by setting the WebApplication attribute of the WebSite, WebDir, and WebVirtualDir elements.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#), [WebSite](#), [WebVirtualDir](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [WebApplicationExtension](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
AllowSessions	YesNoDefaultType	Sets the Enable Session State option. When enabled, you can set the session timeout using the SessionTimeout attribute.	
Buffer	YesNoDefaultType	Sets the option that enables response buffering in the	

		application, which allows ASP script to set response headers anywhere in the script.
ClientDebugging	YesNoDefaultType	Enable ASP client-side script debugging.
DefaultScript	Enumeration	Sets the default script language for the site. This attribute's value should be one of the following: <i>VBScript</i> <i>JScript</i>
Isolation	Enumeration	Sets the application isolation level for this application for pre-IIS 6 applications. This attribute's value should be one of the following: <i>low</i> Means the application executes within the IIS process. <i>medium</i> Executes pooled in a separate process. <i>high</i> Means

execution alone
in a separate
process.

Name	String	Sets the name of this application.	Yes
ParentPaths	YesNoDefaultType	Sets the parent paths option, which allows a client to use relative paths to reach parent directories from this application.	
ScriptTimeout	Integer	Sets the timeout value for executing ASP scripts.	
ServerDebugging	YesNoDefaultType	Enable ASP server-side script debugging.	
SessionTimeout	Integer	Sets the timeout value for sessions in minutes.	
WebAppPool	String	References the Id attribute of a WebAppPool element to use as the application pool for this application in IIS 6 applications.	

See Also

[Wix Schema](#)

WebApplicationExtension Element

Description

Extension for WebApplication

Windows Installer references

None

Parents

[WebApplication](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
CheckPath	YesNoType		
Executable	String	usually a Property that resolves to short file name path	Yes
Extension	String	Extension being registered. Do not prefix with a '.' (e.g. you should use "html", not ".html"). To register for all extensions, use Extension="*". To register a wildcard application map (which handles all requests, even those for directories or files with no extension) omit the Extension attribute completely.	
Script	YesNoType		
Verbs	String		

See Also

Wix Schema

Version 2.0.4820.0

WebAppPool Element

Description

IIS6 Application Pool

Windows Installer references

None

Parents

[Component](#), [Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [RecycleTime](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Id of the AppPool.	Yes
CpuAction	Enumeration	Action taken when CPU exceeds maximum CPU use (as defined with MaxCpuUsage and RefreshCpu). This attribute's value should be one of the following: <i>none</i> <i>shutdown</i>	
Identity	Enumeration	Identity you want the AppPool to run under. Use the 'other' value	

in conjunction with the User attribute to specify non-standard user. This attribute's value should be one of the following:

networkService

localService

localSystem

other

IdleTimeout	Integer	Shutdown worker process after being idle for (time in minutes).	
MaxCpuUsage	PercentType	Maximum CPU usage (percent).	
MaxWorkerProcesses	Integer	Maximum number of worker processes.	
Name	String	Name of the AppPool to be shown in IIs.	Yes
PrivateMemory	Integer	Specifies the amount of private memory (in KB) that a worker process can use before the worker process recycles. The maximum value supported for this attribute is 4,294,967 KB.	
QueueLimit	Integer	Limit the kernel request queue (number of requests).	
RecycleMinutes	Integer	How often, in minutes, you want the	

		AppPool to be recycled.
RecycleRequests	Integer	How often, in requests, you want the AppPool to be recycled.
RefreshCpu	Integer	Refresh CPU usage numbers (in minutes).
User	String	User account to run the AppPool as. To use this, you must set the Identity attribute to 'other'.
VirtualMemory	Integer	Specifies the amount of virtual memory (in KB) that a worker process can use before the worker process recycles. The maximum value supported for this attribute is 4,294,967 KB.

See Also

[Wix Schema](#)

WebDir Element

Description

Defines a subdirectory within an IIS web site. When this element is a child of WebSite, the web directory is defined within that web site. Otherwise the web directory must reference a WebSite element via the WebSite attribute.

Windows Installer references

None

Parents

[Component](#), [Include](#), [WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
DirProperties	String	References the Id attribute for a WebDirProperties element that specifies the security and access properties for this web directory.	Yes
Path	String	Specifies the name of this web directory.	Yes
WebSite	String	References the Id attribute for a WebSite element in which this directory belongs. Required when this element is not a child of a WebSite element.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebDirProperties Element

Description

WebDirProperties used by one or more WebSites. Lists properties common to IIS web sites and vroots. Corresponding properties can be viewed through the IIS Manager snap-in. One property entry can be reused by multiple sites or vroots using the Id field as a reference, using WebVirtualDir.DirProperties, WebSite.DirProperties, or WebDir.DirProperties.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
AccessSSL	YesNoType	A value of true indicates that file access requires SSL file permission processing, with or without a client certificate. This corresponds to AccessSSL flag for AccessSSLFlags IIS metabase property.	
AccessSSL128	YesNoType	A value of true indicates	

that file access requires SSL file permission processing with a minimum key size of 128 bits, with or without a client certificate. This corresponds to AccessSSL128 flag for AccessSSLFlags IIS metabase property.

AccessSSLMapCert	YesNoType	This corresponds to AccessSSLMapCert flag for AccessSSLFlags IIS metabase property.
AccessSSLNegotiateCert	YesNoType	This corresponds to AccessSSLNegotiateCert flag for AccessSSLFlags IIS metabase property.
AccessSSLRequireCert	YesNoType	This corresponds to AccessSSLRequireCert flag for AccessSSLFlags IIS metabase property.
AnonymousAccess	YesNoType	Sets the Enable Anonymous Access checkbox, which maps anonymous users to a Windows user account. When setting this to 'yes' you should also provide the user account using the AnonymousUser attribute, and determine what setting to use for the IISControlledPassword attribute. Defaults to 'no.'
AnonymousUser	String	Reference to the Id attribute on the User

		element to be used as the anonymous user for the directory. See the User element for more information.
AspDetailedError	YesNoType	Sets the option for whether to send detailed ASP errors back to the client on script error. Default is 'no.'
AuthenticationProviders	String	Comma delimited list, in order of precedence, of Windows authentication providers that IIS will attempt to use: NTLM, Kerberos, Negotiate, and others.
BasicAuthentication	YesNoType	Sets the Basic Authentication option, which allows clients to provide credentials in plaintext over the wire. Defaults to 'no.'
CacheControlCustom	String	Custom HTTP 1.1 cache control directives.
CacheControlMaxAge	Integer	Integer value specifying the cache control maximum age value.
ClearCustomError	YesNoType	Specifies whether IIS will return custom errors for this directory.
DefaultDocuments	String	The list of default documents to set for this web directory, in comma-delimited format.
DigestAuthentication	YesNoType	Sets the Digest Authentication option,

which allows using digest authentication with domain user accounts. Defaults to 'no.'

Execute	YesNoType	
HttpExpires	String	Value to set the HttpExpires attribute to for a Web Dir in the metabase.
IIsControlledPassword	YesNoType	Sets whether IIS should control the password used for the Windows account specified in the AnonymousUser attribute. Defaults to 'no.'
Index	YesNoType	Sets the Index Resource option, which specifies whether this web directory should be indexed. Defaults to 'no.'
LogVisits	YesNoType	Sets whether visits to this site should be logged. Defaults to 'no.'
PassportAuthentication	YesNoType	Sets the Passport Authentication option, which allows clients to provide credentials via a .Net Passport account. Defaults to 'no.'
Read	YesNoType	
Script	YesNoType	
WindowsAuthentication	YesNoType	Sets the Windows Authentication option, which enables integrated Windows authentication to be used on the site. Defaults to 'no.'

Write

[YesNoType](#)

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebError Element

Description

Custom Web Errors used by WebSites and Virtual Directories.

Windows Installer references

None

Parents

[WebSite](#), [WebVirtualDir](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
ErrorCode	Integer	HTTP 1.1 error code.	Yes
File	String	File to be sent to the client for this error code and sub code. This can be formatted. For example: [#FileId].	
SubCode	Integer	Error sub code. Set to 0 to get the wild card "*".	Yes
URL	String	URL to be sent to the client for this error code and sub code. This can be formatted.	

Remarks

You can only use error code and sub code combinations which are supported by IIS. Attempting to set a custom error for an error code and sub code combination that is not supported by IIS (in the default list of error codes) will result in an installation failure.

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebFilter Element

Description

IIs Filter for a Component

Windows Installer references

None

Parents

[Component](#), [Include](#), [WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The unique Id for the web filter.	Yes
Description	String	Description of the filter.	
Flags	Integer	Sets the MD_FILTER_FLAGS metabase key for the filter. This must be an integer. See MSDN 'FilterFlags' documentation for more details.	
LoadOrder	String	Allowed values: "first", "last", number	
Name	String	The name of the filter to be used in IIS.	Yes
Path	String	Usually a Property that resolves to short file name path	Yes
WebSite	String	Required if not found as child of WebSite element	

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebLog Element

Description

WebLog definition.

Windows Installer references

None

Parents

[Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the WebLog.	Yes
Type	Enumeration	This attribute's value should be one of the following: <i>IIS</i> Microsoft IIS Log File Format <i>NCSA</i> NCSA Common Log File Format <i>none</i> Disables logging. <i>ODBC</i> ODBC Logging <i>W3C</i> W3C Extended Log File Format	Yes

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebProperty Element

Description

IIS Properties

Windows Installer references

None

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Enumeration	This attribute's value should be one of the following: <i>ETagChangeNumber</i> <i>IIs5IsolationMode</i> <i>MaxGlobalBandwidth</i> <i>LogInUTF8</i>	
Value	String	The value to be used for the WebProperty specified in the Id attribute. See the remarks section for information on acceptable values for each Id.	

Remarks

Here is an explanation of the acceptable values for each property and their meaning:

- For the Ids *IIs5IsolationMode* and *LogInUTF8*, no value should be

specified since the presence of this property indicates that the setting should be set.

- For the MaxGlobalBandwidth Id, the value should be specified in kilobytes. The value should be a base 10 number.
- ETagChangeNumber sets the machine-specific portion of ETag as a number. This value, when synchronized across servers in a web farm, allows the web farm to return an identical ETag for a given resource regardless of the server that handled the request. The value should be a base 10 number.

See Also

[Wix Schema](#)

Version 2.0.4820.0

WebServiceExtension Element

Description

The WebServiceExtension property is used by the Web server to determine whether a Web service extension is permitted to run.

Windows Installer references

None

Parents

[Component](#), [Include](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Allow	YesNoType	Indicates if the extension is allowed or denied.	Yes
Description	String	Description of the extension.	
File	String	Usually a Property that resolves to short file name path	Yes
Group	String	String used to identify groups of extensions.	
UIDeletable	YesNoType	Indicates if the UI is allowed to delete the extension from the list of not. Default: Not deletable.	

See Also

[Wix Schema](#)

WebSite Element

Description

Its Web Site

Windows Installer references

None

Parents

[Component](#), [Fragment](#), [Include](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [WebAddress](#) (min: 1, max: unbounded)
2. [WebApplication](#) (min: 0, max: 1)
3. Choice of elements (min: 0, max: unbounded)
 - [CertificateRef](#) (min: 0, max: unbounded)
 - [HTTPHeader](#) (min: 0, max: unbounded)
 - [WebDir](#) (min: 0, max: unbounded)
 - [WebError](#) (min: 0, max: unbounded)
 - [WebFilter](#) (min: 0, max: unbounded)
 - [WebVirtualDir](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the WebSite. Used within the MSI package only.	Yes
AutoStart	YesNoType	Specifies whether to automatically start the web site.	

ConfigureIfExists	YesNoType	Specifies whether to configure the web site if it already exists. Note: This will not affect uninstall behavior. If the web site exists on uninstall, it will be removed.	
ConnectionTimeout	NonNegativeInteger	Sets the timeout value for connections in seconds.	
Description	String	This is the name of the web site that will show up in the IIS management console.	Yes
Directory	String	Root directory of the web site. Resolved to a directory in the Directory table at install time by the server custom actions.	
DirProperties	String	Reference to WebDirProperties element.	
Sequence	Integer	Sequence that the web site is to be created in.	
StartOnInstall	YesNoType	Specifies whether to start the web site on install.	

WebApplication	String	Reference to a WebApplication that is to be installed as part of this web site.
WebLog	String	Reference to WebLog definition.

Remarks

Nesting WebSite under a Component element will result in a WebSite being installed to the machine as the package is installed.

Nesting WebSite under Product, Fragment, or Module results in a web site "locator" record being created in the IIsWebSite table. This means that the web site itself is neither installed nor uninstalled by the MSI package. It does make the database available for referencing from a WebApplication, WebVirtualDir or WebDir record. This allows an MSI to install WebApplications, WebVirtualDirs or WebDirs to already existing web sites on the machine. The install will fail if the web site does not exist in these cases.

See Also

[Wix Schema](#)

WebVirtualDir Element

Description

Defines an IIS virtual directory. When this element is a child of WebSite element, the virtual directory is defined within that web site. Otherwise this virtual directory must reference a WebSite element via the WebSite attribute

Windows Installer references

None

Parents

[Component](#), [Include](#), [WebSite](#), [WebVirtualDir](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [WebApplication](#) (min: 0, max: 1)
2. [WebError](#) (min: 0, max: unbounded)
3. [WebVirtualDir](#) (min: 0, max: unbounded)
4. [HttpHeader](#) (min: 0, max: unbounded)
5. [MimeMap](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
Alias	String	Sets the application name, which is the URL relative path used to access this virtual directory	Yes
Directory	String	References the Id attribute for a Directory element that points to the content for this virtual directory.	Yes

DirProperties	String	References the Id attribute for a WebDirProperties element that specifies the security and access properties for this virtual directory.
WebApplication	String	References the Id attribute for a WebApplication element that specifies web application settings for this virtual directory. If a WebApplication child is not specified, the virtual directory does not host web applications.
WebSite	String	References the Id attribute for a WebSite in which this virtual directory belongs. Required when this element is not a child of WebSite element.

See Also

[Wix Schema](#)

Wix Element

Description

This is the top-level container element for every wxs file. Amongst the possible children, the Product, Module, and PatchCreation elements are analogous to the main function in a C program. There can only be one of these present when linking occurs. Product compiles into an msi file, Module compiles into an msm file, PatchCreation compiles into a pcg file. The Fragment element is an atomic unit which ultimately links into either a Product, Module, or PatchCreation. The Fragment can either be completely included or excluded during linking.

Windows Installer references

None

Parents

None

Inner Text

None

Children

Choice of elements (min: 0, max: 1)

- [PatchCreation](#) (min: 0, max: 1)
- Sequence (min: 1, max: 1)
 1. Choice of elements (min: 0, max: 1)
 - [Module](#) (min: 0, max: 1)
 - [Product](#) (min: 0, max: 1)
 2. [Fragment](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
RequiredVersion	VersionType	Required version of the WiX toolset to compile this input file.	

See Also

[Wix Schema](#)

Version 2.0.4820.0

WriteEnvironmentStrings Element

Description

Modifies the values of environment variables. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

WriteIniValues Element

Description

Writes the .ini file information that the application needs written to its .ini files. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

WriteRegistryValues Element

Description

Sets up an application's registry information. The condition for this action may be specified in the element's inner text.

Windows Installer references

None

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

XmlFile Element

Description

Adds or removes .xml file entries. If you use the XmlFile element you must link with wixca.wixlib because it requires the XmlFile custom actions.

Windows Installer references

None

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for xml file modification.	Yes
Action	Enumeration	The type of modification to be made to the XML file when the component is installed. This attribute's value should be one of the following: <i>createElement</i> Creates a new element under the element specified in ElementPath. The Name attribute is required in this case and specifies the name	Yes

of the new element. The Value attribute is not necessary when createElement is specified as the action. If the Value attribute is set, it will cause the new element's text value to be set.

deleteValue

Deletes a value from the element specified in the ElementPath. If Name is specified, the attribute with that name is deleted. If Name is not specified, the text value of the element specified in the ElementPath is deleted. The Value attribute is ignored if deleteValue is the action specified.

setValue

Sets a value in the element specified in the ElementPath. If Name is specified, and attribute with that name is set to the value specified in Value. If Name is not specified, the text value of the element is set. Value is a required attribute if setValue is the action specified.

CreateElement [YesNoType](#) Specifies whether or not to

		create an Element with the name specified in the Name attribute.	
ElementPath	String	The XPath of the element to be modified. Note that this is a formatted field and therefore, square brackets in the XPath must be escaped.	Yes
File	String	Path of the .xml file to configure.	Yes
Name	String	Name of XML node to set/add to the specified element. Not setting this attribute causes the element's text value to be set. Otherwise this specified the attribute name that is set.	
Permanent	YesNoType	Specifies whether or not the modification should be removed on uninstall. This has no effect on uninstall if the action was deleteValue.	
Sequence	Integer	Specifies the order in which the modification is to be attempted on the XML file. It is important to ensure that new elements are created before you attempt to add an attribute to them.	
Value	String	The value to be written.	

See Also
[Wix Schema](#)

Mmc Schema

Copyright (c) Microsoft Corporation. All rights reserved. The use and distribution terms for this software are covered by the Common Public License 1.0 (<http://opensource.org/licenses/cpl.php>) which can be found in the file CPL.TXT at the root of this distribution. By using this software in any fashion, you are agreeing to be bound by the terms of this license. You must not remove this notice, or any other, from this software.

The source code schema for the Windows Installer XML Toolset MMC Extension.

Target Namespace

<http://schemas.microsoft.com/wix/MmcExtension>

All Elements

- [ExtendedNodeType](#)
- [PublishedNodeType](#)
- [Resources](#)
- [SnapIn](#)

ExtendedNodeType Element

Description

Published node type that is extended by this snap-in.

Windows Installer references

None

Parents

[SnapIn](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Uuid	The guid representing the extended node.	Yes
Description	String	The description of the extension.	

See Also

[Mmc Schema](#)

PublishedNodeType Element

Description

Published node type that can be extended by extension snap-ins.

Windows Installer references

None

Parents

[SnapIn](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Uuid	The guid representing the extensible node.	Yes
Description	String	The description of the extensible node.	

See Also

[Mmc Schema](#)

Resources Element

Description

Element describing the localized resources for this snap-in.

Windows Installer references

None

Parents

[SnapIn](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
DescriptionId	Integer	The resource ID for the description of the snap-in in the resources DLL.	
DisplayNameId	Integer	The resource ID for the display name of the snap-in in the resources DLL.	
DllName	String	The name of the DLL containing the embedded resources for this snap-in.	Yes
FolderBitmapsColorMask	Integer	The color mask for transparency in folder bitmaps.	

IconId	Integer	The resource ID for the icon of the snap-in in the resources DLL. Used for the icon of a saved MSC file, and the icon in the top left of the MMC window, not for the snap-in selection dialog.
LargeFolderBitmapId	Integer	The resource ID for the large folder bitmap of the snap-in in the resources DLL. Used for the snap-in selection dialog when Add/Remove Snap-ins is chosen.
SmallFolderBitmapId	Integer	The resource ID for the small folder bitmap of the snap-in in the resources DLL. Used for the snap-in selection dialog when Add/Remove Snap-ins is chosen.
SmallFolderSelectedBitmapId	Integer	The resource ID for the small selected folder bitmap of the snap-in in the resources DLL. Used for the snap-in selection dialog

when Add/Remove Snap-ins is chosen.

VendorId	Integer	The resource ID for the vendor of the snap-in in the resources DLL.
VersionId	Integer	The resource ID for the version of the snap-in in the resources DLL.

See Also

[Mmc Schema](#)

Version 2.0.4820.0

SnapIn Element

Description

A managed MMC snap-in, with optional published extendible nodes.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [ExtendedNodeType](#) (min: 0, max: unbounded): Node type of another snap-in that is extended by this snap-in.
- [PublishedNodeType](#) (min: 0, max: unbounded): Published node types that can be extended by extension snap-ins.
- [Resources](#) (min: 0, max: unbounded): Element describing the localized resources for this snap-in.

Attributes

Name	Type	Description	Required
Id	Uuid	The guid representing the snap-in's identity.	Yes
About	Uuid	The guid representing the snap-in's help topic. Defaults to {00000000-0000-0000-0000-000000000000}.	
AssemblyName	String	The name of the	

		assembly in which the snap-in is defined.	
ClassType	String	The fully-qualified type name of the snap-in.	Yes
DefaultCulture	String	The culture of the snap-in assembly. Defaults to neutral.	
DefaultPublicKeyToken	String	The public key token of the snap-in. Defaults to null.	
DefaultVersion	String	The version of the snap-in assembly. Defaults to 1.0.0.0.	
Description	String	The description of the snap-in, which will be shown to users in the Add/Remove snap-in dialog.	
ExtensionType	Enumeration	Specifies the type of the extension. This attribute's value should be one of the following: <i>ContextMenu</i> <i>NameSpace</i> <i>PropertySheet</i> <i>Task</i> <i>ToolBar</i> <i>View</i>	
MmcVersion	String	The version of MMC that this snap-in was	

		compiled to. Defaults to 3.0.0.0.	
Name	String	The name of the snap-in as shown to users in the Add/Remove snap-in dialog.	Yes
Provider	String	The provider of the snap-in as shown to users in the Add/Remove snap-in dialog.	
RuntimeVersion	String	The version of the CLR that this snap-in was compiled to. Defaults to 2.0.50727.	

See Also

[Mmc Schema](#)

Version 2.0.4820.0

Netfx Schema

Copyright (c) Microsoft Corporation. All rights reserved. The use and distribution terms for this software are covered by the Common Public License 1.0 (<http://opensource.org/licenses/cpl.php>) which can be found in the file CPL.TXT at the root of this distribution. By using this software in any fashion, you are agreeing to be bound by the terms of this license. You must not remove this notice, or any other, from this software.

The source code schema for the Windows Installer XML Toolset .NET Framework Extension.

Target Namespace

<http://schemas.microsoft.com/wix/NetFxExtension>

All Elements

- [NativeImage](#)

NativeImage Element

Description

Improves the performance of managed applications by creating native images. Requires the .NET Framework 2.0 to be installed on the target machine since it runs [NGen](#).

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The identifier for this NativeImage.	Yes
AppBaseDirectory	String	The identifier of the directory to use for locating dependent assemblies. For DLL assemblies and assemblies installed to the GAC, this attribute should be set to the directory of the application which loads this assembly. For EXE assemblies, this attribute does not need to be set because NGen will use the directory of the assembly file by default.	

AssemblyApplication String

The identifier of the application which will load this assembly. For DLL assemblies which are loaded via reflection, this attribute should be set to indicate the application which will load this assembly. The configuration of the application (usually specified via an exe.config file) will be used to determine how to resolve dependencies for this assembly. When a shared component is loaded at run time, using the Load method, the application's configuration file determines the dependencies that are loaded for the shared component — for example, the version of a dependency that is loaded. This attribute gives guidance on which dependencies would be loaded at run time in order to figure out which dependency assemblies will also need to have native images generated (assuming the Dependency attribute is not set to "no"). This attribute cannot be set if the AssemblyApplication

attribute is set on the parent File element (please note that these attributes both refer to the same application assembly but do very different things: specifying File/@AssemblyApplication will force an assembly to install to a private location next to the indicated application, whereas this AssemblyApplication attribute will be used to help resolve dependent assemblies while generating native images for this assembly).

Debug	YesNoType	Set to "yes" to generate native images that can be used under a debugger. The default value is "no".
Dependencies	YesNoType	Set to "no" to generate the minimum number of native images. The default value is "yes".
Platform	Enumeration	Sets the platform(s) for which native images will be generated. This attribute's value should be one of the following: <i>32bit</i> Generate native images only for the 32-bit version of the .NET Framework on the target machine. This is the default

value.

64bit

Generate native images only for the ia64 or x86 version of the .NET Framework on the target machine. If no 64-bit .NET Framework is available on the target machine, attempting to generate native images will fail.

all

Generate native images for all platforms of the .NET Framework available on the target machine.

Priority	Enumeration	Sets the priority of generating the native images for this assembly. This attribute's value should be one of the following: <i>0</i> This is the highest priority, it means that image generation occurs synchronously during the setup process. This option will slow down setup performance. <i>1</i> This will queue image
----------	-------------	---

generation to the NGen service to occur immediately. This option will slow down setup performance.

2

This will queue image generation to the NGen service to occur after all priority 1 assemblies have completed. This option will slow down setup performance.

3

This is the lowest priority, it will queue image generation to occur when the machine is idle. This option should not slow down setup performance. This is the default value.

Profile	YesNoType	Set to "yes" to generate native images that can be used under a profiler. The default value is "no".
---------	---------------------------	--

Remarks

Native images are files containing compiled processor-specific machine code, which are installed into the native image cache on the local computer. The runtime can use native images from the cache instead using the just-in-time (JIT) compiler to compile the original assembly.

See Also

[Netfx Schema](#)

Version 2.0.4820.0

Vs Schema

Copyright (c) Microsoft Corporation. All rights reserved. The use and distribution terms for this software are covered by the Common Public License 1.0 (<http://opensource.org/licenses/cpl.php>) which can be found in the file CPL.TXT at the root of this distribution. By using this software in any fashion, you are agreeing to be bound by the terms of this license. You must not remove this notice, or any other, from this software.

The source code schema for the Windows Installer XML Toolset Visual Studio Extension.

Target Namespace

<http://schemas.microsoft.com/wix/VSEExtension>

All Elements

- [HelpCollection](#)
- [HelpCollectionRef](#)
- [HelpFile](#)
- [HelpFileRef](#)
- [HelpFilter](#)
- [HelpFilterRef](#)
- [PlugCollectionInto](#)

HelpCollection Element

Description

Help Namespace for a help collection. The parent file is the key for the HxC (Collection) file.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [HelpFileRef](#) (min: 0, max: unbounded)
- [HelpFilterRef](#) (min: 0, max: unbounded)
- [PlugCollectionInto](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpNamespace.	Yes
Description	String	Friendly name for Namespace.	
Name	String	Internal Microsoft Help ID for this Namespace.	Yes

See Also

[Vs Schema](#)

HelpCollectionRef Element

Description

Create a reference to a HelpCollection element in another Fragment.

Windows Installer references

None

Parents

[Fragment](#), [Product](#)

Inner Text

None

Children

Choice of elements (min: 0, max: unbounded)

- [HelpFileRef](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpNamespace Table.	Yes

See Also

[Vs Schema](#)

HelpFile Element

Description

File for Help Namespace. The parent file is the key for HxS (Title) file.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpFile Table.	Yes
AttributeIndex	String	Key for HxR (Attributes) file.	
Index	String	Key for HxI (Index) file.	
Language	Integer	Language ID for content file.	
Name	String	Internal Microsoft Help ID for this HelpFile.	Yes
SampleLocation	String	Key for a file that is in the "root" of the samples directory for this HelpFile.	
Search	String	Key for HxQ (Query) file.	

See Also

[Vs Schema](#)

HelpFileRef Element

Description

Create a reference to a HelpFile element in another Fragment.

Windows Installer references

None

Parents

[HelpCollection](#), [HelpCollectionRef](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpFile Table.	Yes

See Also

[Vs Schema](#)

Version 2.0.4820.0

HelpFilter Element

Description

Filter for Help Namespace.

Windows Installer references

None

Parents

[Fragment](#), [Product](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpFilter.	Yes
FilterDefinition	String	Query String for Help Filter.	
Name	String	Friendly name for Filter.	Yes

See Also

[Vs Schema](#)

HelpFilterRef Element

Description

Create a reference to a HelpFile element in another Fragment.

Windows Installer references

None

Parents

[HelpCollection](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpFilter.	Yes

See Also

[Vs Schema](#)

Version 2.0.4820.0

PlugCollectionInto Element

Description

Plugin for Help Namespace.

Windows Installer references

None

Parents

[HelpCollection](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Attributes	String	Key for HxA (Attributes) file of child namespace.	
TableOfContents	String	Key for HxT file of child namespace.	
TargetCollection	String	Foriegn Key into HelpNamespace table for the parent namespace into which the child will be inserted.	Yes
TargetTableOfContents	String	Key for HxT file of parent namespace that now includes the new child namespace.	

See Also

[Vs Schema](#)

Pubca Schema

Copyright (c) Microsoft Corporation. All rights reserved.

The use and distribution terms for this software are covered by the Common Public License 1.0 (<http://opensource.org/licenses/cpl.php>) which can be found in the file CPL.TXT at the root of this distribution. By using this software in any fashion, you are agreeing to be bound by the terms of this license.

You must not remove this notice, or any other, from this software.

Schema for describing standard actions in the Windows Installer.

Root Elements

- [ComPlusPartition](#)
- [MessageQueue](#)

Target Namespace

<http://schemas.microsoft.com/wix/2005/02/pubca>

Document Should Look Like

- `<?xml version="1.0"?>`
`<ComPlusPartition`
`xmlns="http://schemas.microsoft.com/wix/2005/02/pubca">`
`.`
`.`
`.`
`</ComPlusPartition>`
- `<?xml version="1.0"?>`
`<MessageQueue`
`xmlns="http://schemas.microsoft.com/wix/2005/02/pubca">`
`.`
`.`
`.`
`</MessageQueue>`

All Elements

- [ComPlusApplication](#)
- [ComPlusApplicationRole](#)
- [ComPlusAssembly](#)
- [ComPlusAssemblyDependency](#)
- [ComPlusComponent](#)
- [ComPlusGroupInApplicationRole](#)
- [ComPlusGroupInPartitionRole](#)
- [ComPlusInterface](#)
- [ComPlusMethod](#)
- [ComPlusPartition](#)
- [ComPlusPartitionRole](#)
- [ComPlusPartitionUser](#)
- [ComPlusRoleForComponent](#)
- [ComPlusRoleForInterface](#)
- [ComPlusRoleForMethod](#)
- [ComPlusSubscription](#)
- [ComPlusUserInApplicationRole](#)
- [ComPlusUserInPartitionRole](#)
- [MessageQueue](#)
- [MessageQueuePermission](#)

ComPlusApplication Element

Description

Defines a COM+ application. If this element is a descendent of a Component element, the application will be created in association with this component. If the element is a child of any of the Fragment, Module or Product elements it is considered to be a locator, referencing an existing application.

If the element is a child of a ComPlusPartition element, or have its Partition attribute set, the application will be installed under the referenced partition.

Windows Installer references

None

Parents

[ComPlusPartition](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusApplicationRole](#) (min: 0, max: unbounded)
 - [ComPlusAssembly](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
Id	String	Identifier for the element
AccessChecksLevel	Enumeration	This attribute's value should be one of the following: <i>applicationLevel</i> <i>applicationComponent</i>

Activation	Enumeration	This attribute's value should be one of the following: <i>inproc</i> <i>local</i>
ApplicationAccessChecksEnabled	YesNoType	
ApplicationDirectory	String	
ApplicationId	String	Id for the application. attribute can be omit which case an id will generated on install. element is a locator, attribute can be omit value is provided for Name attribute.
Authentication	Enumeration	This attribute's value should be one of the following: <i>default</i> <i>none</i> <i>connect</i> <i>call</i> <i>packet</i> <i>integrity</i> <i>privacy</i>
AuthenticationCapability	Enumeration	This attribute's value should be one of the following: <i>none</i> <i>secureReference</i> <i>staticCloaking</i>

dynamicCloaking

Changeable	YesNoType	
CommandLine	String	
ConcurrentApps	Int	
CreatedBy	String	
CRMEnabled	YesNoType	
CRMLogFile	String	
Deleteable	YesNoType	
Description	String	
DumpEnabled	YesNoType	
DumpOnException	YesNoType	
DumpOnFailfast	YesNoType	
DumpPath	String	
EventsEnabled	YesNoType	
Identity	String	
ImpersonationLevel	Enumeration	This attribute's value should be one of the following: <i>anonymous</i> <i>identify</i> <i>impersonate</i> <i>delegate</i>
IsEnabled	YesNoType	
MaxDumpCount	Int	
Name	String	Name of the application This attribute can be omitted if the element locator, and a value is provided for the Part attribute.
Partition	String	If the element is not a part of a ComPlusPartition

element, this attribute
 be provided with the
 ComPlusPartition ele
 representing the part
 the application belon

Password	String	
QCAuthenticateMsgs	Enumeration	This attribute's value should be one of the following: <i>secureApps</i> <i>off</i> <i>on</i>
QCListenerMaxThreads	Int	
QueueListenerEnabled	YesNoType	
QueuingEnabled	YesNoType	
RecycleActivationLimit	Int	
RecycleCallLimit	Int	
RecycleExpirationTimeout	Int	
RecycleLifetimeLimit	Int	
RecycleMemoryLimit	Int	
Replicable	YesNoType	
RunForever	YesNoType	
ShutdownAfter	Int	
SoapActivated	YesNoType	
SoapBaseUrl	String	
SoapMailTo	String	
SoapVRoot	String	
SRPEnabled	YesNoType	
SRPTrustLevel	Enumeration	This attribute's value should be one of the following: <i>disallowed</i>

fullyTrusted

ThreeGigSupportEnabled	String
------------------------	--------

See Also

[Pubca Schema](#)

Version 2.0.4820.0

ComPlusApplicationRole Element

Description

Defines an application role. If this element is a descendent of a Component element, the application role will be created in association with this component. If the element is a child of any of the Fragment, Module or Product elements it is considered to be a locator, referencing an existing application role.

Windows Installer references

None

Parents

[ComPlusApplication](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusGroupInApplicationRole](#) (min: 0, max: unbounded)
 - [ComPlusUserInApplicationRole](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Application	String	If the element is not a child of a ComPlusApplication element, this attribute should be provided with the id of a ComPlusApplication element representing the application the role belongs to.	
Description	String		
Name	String	Name of the application role.	Yes

See Also

[Pubca Schema](#)

Version 2.0.4820.0

ComPlusAssembly Element

Description

Represents a DLL or assembly to be registered with COM+. If this element is a child of a ComPlusApplication element, the assembly will be registered in this application. Other ways the Application attribute must be set to an application. The element must be a descendent of a Component element, it can not be a child of a ComPlusApplication locator element.

Windows Installer references

None

Parents

[ComPlusApplication](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusAssemblyDependency](#) (min: 0, max: unbounded)
 - [ComPlusComponent](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Application	String	If the element is not a child of a ComPlusApplication element, this attribute should be provided with the id of a ComPlusApplication element representing the	

application the assembly is to be registered in. This attribute can be omitted for a .NET assembly even if the application is not a child of a ComPlusApplication element.

AssemblyName	String	The name of the assembly used to identify the assembly in the GAC. This attribute can be provided only if DllPathFromGAC is set to "yes".
DllPath	String	The path to locate the assembly DLL during registration. This attribute should be provided if DllPathFromGAC is not set to "yes".
DllPathFromGAC	YesNoType	Indicates that the DLL path should be extracted from the GAC instead for being provided in the DllPath attribute. If this attribute is set to "yes", the name of the assembly can be provided using the AssemblyName attribute. Or, if this AssemblyName attribute is missing, the name will be extracted from the MsiAssemblyName table using the id of the parent Component element.
EventClass	YesNoType	Indicates that the

assembly is to be installed as an event class DLL. This attribute is only valid for native assemblies. The assembly will be installed with the COM+ catalog's InstallEventClass() function.

PSDIPath	String	An optional path to an external proxy/stub DLL for the assembly.	
RegisterInCommit	YesNoType	Indicates that the assembly should be installed in the commit custom action instead of the normal deferred custom action. This is necessary when installing .NET assemblies to the GAC in the same installation, as the assemblies are not visible in the GAC until after the InstallFinalize action has run.	
TlbPath	String	An optional path to an external type lib for the assembly. This attribute must be provided if the Type attribute is set to ".net".	
Type	Enumeration	This attribute's value should be one of the following: <i>native</i> <i>.net</i>	Yes

Remarks

When installing a native assembly, all components contained in the assembly must be represented as ComPlusComponent elements under this element. Any component not listed will not be removed during uninstall.

The fields DllPath, TlbPath and PSDllPath are formatted fields that should contain file paths to their respective file types. A typical value for DllPath for example, should be something like “[#MyAssembly_dll]”, where “MyAssembly_dll” is the key of the dll file in the File table.

Warning: The assembly name provided in the AssemblyName attribute must be a fully specified assembly name, if a partial name is provided a random assembly matching the partial name will be selected.

See Also

[Pubca Schema](#)

ComPlusAssemblyDependency Element

Description

Defines a dependency between two assemblies. This element affects the order in which assemblies are registered. Any assemblies referenced by this element are guaranteed to be registered before, and unregistered after, the assembly referenced by the parent ComPlusAssembly element.

Windows Installer references

None

Parents

[ComPlusAssembly](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
RequiredAssembly	String	Reference to the id of the assembly required by the parent ComPlusAssembly element.	Yes

Remarks

It is only necessary to explicitly specify dependencies between assemblies contained in the same package (MSI or MSM). Assemblies merged in to a package from a merge module will always be installed before any assemblies specified in the base package. Assemblies merged in from different merge modules are sequenced using the ModuleDependency MSI table. It is not possible to have cross dependencies between merge modules or have an assembly in a merge module depend on an assembly in the

base package.

See Also

[Pubca Schema](#)

Version 2.0.4820.0

ComPlusComponent Element

Description

Represents a COM+ component in an assembly.

Windows Installer references

None

Parents

[ComPlusAssembly](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusInterface](#) (min: 0, max: unbounded)
 - [ComPlusRoleForComponent](#) (min: 0, max: unbounded)
 - [ComPlusSubscription](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
Id	String	Identifier for the element.
AllowInprocSubscribers	YesNoType	
CLSID	Uuid	CLSID of the component.
ComponentAccessChecksEnabled	YesNoType	
ComponentTransactionTimeout	YesNoType	
ComponentTransactionTimeoutEnabled	YesNoType	
COMIntrinsics	YesNoType	
ConstructionEnabled	YesNoType	
ConstructorString	String	

CreationTimeout	Int	
Description	String	
EventTrackingEnabled	YesNoType	
ExceptionClass	String	
FireInParallel	YesNoType	
IISIntrinsics	YesNoType	
InitializesServerApplication	YesNoType	
IsEnabled	YesNoType	
IsPrivateComponent	YesNoType	
JustInTimeActivation	YesNoType	
LoadBalancingSupported	YesNoType	
MaxPoolSize	Int	
MinPoolSize	Int	
MultiInterfacePublisherFilterCLSID	String	
MustRunInClientContext	YesNoType	
MustRunInDefaultContext	YesNoType	
ObjectPoolingEnabled	YesNoType	
PublisherID	String	
SoapAssemblyName	String	
SoapTypeName	String	
Synchronization	Enumeration	This attribute's value should be one of the following: <i>ignored</i> <i>none</i> <i>supported</i> <i>required</i> <i>requiresNew</i>
Transaction	Enumeration	This attribute's value should be

one of the following:
ignored
none
supported
required
requiresNew

TxIsolationLevel

Enumeration

This attribute's value should be one of the following:
any
readUnCommit
readCommitted
repeatableRead
serializable

See Also

[Pubca Schema](#)

ComPlusGroupInApplicationRole Element

Description

This element represents a security group membership in an application role. When the parent component of this element is installed, the user will be added to the associated application role. This element must be a descendent of a Component element; it can not be a child of a ComPlusApplicationRole locator element. To reference a locator element use the ApplicationRole attribute.

Windows Installer references

None

Parents

[ComPlusApplicationRole](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	If the element is not a child of a ComPlusApplicationRole element, this attribute should be provided with the id of a ComPlusApplicationRole element representing the application role the user is to be added to.	
Group	String	Foreign key into the Group table.	Yes

See Also

Pubca Schema

Version 2.0.4820.0

ComPlusGroupInPartitionRole Element

Description

This element represents a security group membership in a partition role. When the parent component of this element is installed, the security group will be added to the associated partition role.

Windows Installer references

None

Parents

[ComPlusPartitionRole](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Group	String	Foreign key into the Group table.	Yes
PartitionRole	String	The id of a ComPlusPartitionRole element representing the partition the user should be added to.	

See Also

[Pubca Schema](#)

ComPlusInterface Element

Description

Represents an interface for a COM+ component.

Windows Installer references

None

Parents

[ComPlusComponent](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusMethod](#) (min: 0, max: unbounded)
 - [ComPlusRoleForInterface](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Description	String		
IID	Uuid	IID of the interface.	Yes
QueuingEnabled	YesNoType		

See Also

[Pubca Schema](#)

ComPlusMethod Element

Description

Represents a method for an interface.

Windows Installer references

None

Parents

[ComPlusInterface](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ComPlusRoleForMethod](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
AutoComplete	YesNoType		
Description	String		
Index	Int	Dispatch id of the method. If this attribute is not set a value must be provided for the Name attribute.	
Name	String	Name of the method. If this attribute is not set a value must be provided for the Index attribute.	

See Also

[Pubca Schema](#)

ComPlusPartition Element

Description

Defines a COM+ partition. If this element is a child of a Component element, the partition will be created in association with this component. If the element is a child of any of the Fragment, Module or Product elements it is considered to be a locator, referencing an existing partition.

Windows Installer references

None

Parents

None

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusApplication](#) (min: 0, max: unbounded)
 - [ComPlusPartitionRole](#) (min: 0, max: unbounded)
 - [ComPlusPartitionUser](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Changeable	YesNoType		
Deleteable	YesNoType		
Description	String		
Name	String	Name of the partition. This attribute can be omitted if the element is a locator, and a value is provided for the PartitionId	

		attribute.
PartitionId	String	Id for the partition. This attribute can be omitted, in which case an id will be generated on install. If the element is a locator, this attribute can be omitted if a value is provided for the Name attribute.

See Also

[Pubca Schema](#)

Version 2.0.4820.0

ComPlusPartitionRole Element

Description

Defines a COM+ partition role. Partition roles can not be created; this element can only be used as a locator to reference an existing role.

Windows Installer references

None

Parents

[ComPlusPartition](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusGroupInPartitionRole](#) (min: 0, max: unbounded)
 - [ComPlusUserInPartitionRole](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Name	String	Name of the partition role.	Yes
Partition	String	The id of a ComPlusPartition element representing the partition the role belongs to.	

See Also

[Pubca Schema](#)

ComPlusPartitionUser Element

Description

Represents a default partition definition for a user. When the parent component of this element is installed, the default partition of the user will be set to the referenced partition.

Windows Installer references

None

Parents

[ComPlusPartition](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Partition	String	The id of a ComPlusPartition element representing the partition that will be the default partition for the user.	
User	String	Foreign key into the User table.	Yes

See Also

[Pubca Schema](#)

ComPlusRoleForComponent Element

Description

Represents a role assignment to a COM+ component.

Windows Installer references

None

Parents

[ComPlusComponent](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	Id of the ComPlusApplicationRole element representing the role that shall be granted access to the component.	Yes
Component	String	If the element is not a child of a ComPlusComponent element, this attribute should be provided with the id of a ComPlusComponent element representing the component the role is to be added to.	

See Also

[Pubca Schema](#)

ComPlusRoleForInterface Element

Description

Represents a role assignment to an interface.

Windows Installer references

None

Parents

[ComPlusInterface](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	Id of the ComPlusApplicationRole element representing the role that shall be granted access to the interface.	Yes
Interface	String	If the element is not a child of a ComPlusInterface element, this attribute should be provided with the id of a ComPlusInterface element representing the interface the role is to be added to.	

See Also

[Pubca Schema](#)

ComPlusRoleForMethod Element

Description

Represents a role assignment to a COM+ method.

Windows Installer references

None

Parents

[ComPlusMethod](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	Id of the ComPlusApplicationRole element representing the role that shall be granted access to the method.	Yes
Method	String	If the element is not a child of a ComPlusMethod element, this attribute should be provided with the id of a ComPlusMethod element representing the method the role is to be added to.	

See Also

[Pubca Schema](#)

ComPlusSubscription Element

Description

Defines an event subscription for a COM+ component.

Windows Installer references

None

Parents

[ComPlusComponent](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Component	String	If the element is not a child of a ComPlusComponent element, this attribute should be provided with the id of a ComPlusComponent element representing the component the subscription is to be created for.	
Description	String		
Enabled	YesNoType		
EventClassPartitionID	String		
EventCLSID	String	CLSID of the event	

class for the subscription. If a value for this attribute is not provided, a value for the PublisherID attribute must be provided.

FilterCriteria	String		
InterfaceID	String		
MachineName	String		
MethodName	String		
Name	String	Name of the subscription.	Yes
PerUser	YesNoType		
PublisherID	String	Publisher id for the subscription. If a value for this attribute is not provided, a value for the EventCLSID attribute must be provided.	
Queued	YesNoType		
SubscriberMoniker	String		
SubscriptionId	String	Id of the subscription. If a value is not provided for this attribute, an id will be generated during installation.	
UserName	String		

See Also

[Pubca Schema](#)

ComPlusUserInApplicationRole Element

Description

This element represents a user membership in an application role. When the parent component of this element is installed, the user will be added to the associated application role. This element must be a descendent of a Component element; it can not be a child of a ComPlusApplicationRole locator element. To reference a locator element use the ApplicationRole attribute.

Windows Installer references

None

Parents

[ComPlusApplicationRole](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	If the element is not a child of a ComPlusApplicationRole element, this attribute should be provided with the id of a ComPlusApplicationRole element representing the application role the user is to be added to.	
User	String	Foreign key into the User table.	Yes

See Also

Pubca Schema

Version 2.0.4820.0

ComPlusUserInPartitionRole Element

Description

This element represents a user membership in a partition role. When the parent component of this element is installed, the user will be added to the associated partition role.

Windows Installer references

None

Parents

[ComPlusPartitionRole](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
PartitionRole	String	The id of a ComPlusPartitionRole element representing the partition the user should be added to.	
User	String	Foreign key into the User table.	Yes

See Also

[Pubca Schema](#)

MessageQueue Element

Description

None

Windows Installer references

None

Parents

None

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [MessageQueuePermission](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String		Yes
Authenticate	YesNoType	Default: No.	
BasePriority	Integer		
Component	String		
Journal	YesNoType	Default: No.	
JournalQuota	Integer		
Label	String		Yes
MulticastAddress	String		
PathName	String		Yes
PrivLevel	Enumeration	This attribute's value should be one of the following: <i>none</i> <i>body</i>	

optional

Quota	Integer		
ServiceTypeGuid	String		
Transactional	YesNoType	Default: No.	

See Also

[Pubca Schema](#)

Version 2.0.4820.0

MessageQueuePermission Element

Description

None

Windows Installer references

None

Parents

[MessageQueue](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
ChangeQueuePermissions	YesNoType		
DeleteJournalMessage	YesNoType		
DeleteMessage	YesNoType		
DeleteQueue	YesNoType		
GetQueuePermissions	YesNoType		
GetQueueProperties	YesNoType		
Group	String		
MessageQueue	String		
PeekMessage	YesNoType		
QueueGenericAll	YesNoType		
QueueGenericExecute	YesNoType		
QueueGenericRead	YesNoType		
QueueGenericWrite	YesNoType		
ReceiveJournalMessage	YesNoType		

ReceiveMessage	YesNoType	
SetQueueProperties	YesNoType	
TakeQueueOwnership	YesNoType	
User	String	
WriteMessage	YesNoType	

See Also

[Pubca Schema](#)

Version 2.0.4820.0

Tools

[Preprocessor](#)

[Compiler \(candle\)](#)

[Linker \(light\)](#)

[Decompiler \(dark\)](#)

Preprocessor

Introduction

Often you will need to add different pieces of your setup during build time depending on many factors such as the SKU being built. This is done by using conditional statements that will filter the xml before it is sent to the WiX compiler (candle). If the statement evaluates to true, the block of xml will be sent to candle. If the statement evaluates to false, candle will never see that section of xml.

The conditional statements are Boolean expressions based on environment variables, variables defined in the xml, literal values, and more.

Example

Let's start with an example. Say you want to include a file if you're building the "Enterprise SKU." Your build uses an environment variable %MySku%=Enterprise to specify this sku.

When you build the enterprise sku, this file will be included in the xml passed on to candle. When you build a different sku, the xml from EnterpriseFeature.wxs will be ignored.

```
<?if $(env.MySku) = Enterprise ?>  
  <?include EnterpriseFeature.wxs ?>  
<?endif ?>
```

Include Files <?include?>

As shown in the example above, files can be included by using the include tag. The filename referenced in the tag will be processed as if it were part of this file.

The root element of the include file must be <Include>. There are no other requirements beyond the expected wix schema. For example,

```
<Include>
  <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
    <ComponentRef Id='MyComponent' />
  </Feature>
</Include>
```

Variables

Any variable can be tested for its value or simply its existence. Custom variables can also be defined in your xml.

Three types of variables are supported:

\$(env._NtPostBld)

Gets the environment variable %_NtPostBld%

\$(sys.CurrentDir)

Gets the system variable for the current directory

\$(var.A)

Gets the variable A that was defined in this xml

The preprocessor evaluates variables throughout the entire document, including in <?if?> expressions and attribute values.

Environment Variables

Any environment variable can be referenced with the syntax \$(env.VarName). For example, if you want to retrieve the environment variable %_BuildArch%, you would use \$(env._BuildArch). Environment variable names are case-insensitive.

System Variables

WiX has some built-in variables. They are referenced with the syntax \$(sys.VARNAME) and are always in upper case.

CURRENTDIR - the current directory where the build process is running

SOURCEFILEPATH – the full path to the file being processed

SOURCEFILEDIR – the directory containing the file being processed

NOTE: All built-in directory variables are “\” terminated.

Custom variables <? define ?>

If you want to define custom variables, you need to use the <?define?>

statement. Later, the variables are referred to in the `<?if?>` statements with the syntax `$(var.VarName)`. Variable names are case-sensitive.

How to define the existence of a variable:

```
<?define MyVariable ?>
```

How to define the value of a variable (Quotes are only required if it contains spaces):

```
<?define MyVariable = "Hello World" ?>
```

The right side of the definition can also refer to another variable:

```
<?define MyVariable = $(var.BuildPath)\x86\bin\ ?>
```

How to undefine a variable:

```
<?undef MyVariable ?>
```

Conditional Statements

There are several conditional statements, they include:

`<?if ?>`

`<?ifdef ?>`

`<?ifndef ?>`

`<?else?>`

`<?elseif ?>`

`<?endif?>`

The purpose of the conditional statement is to allow you to include or exclude a segment of xml at build time. If the expression evaluates to true, it will be included. If it evaluates to false, it will be ignored.

The conditional statements always begin with either the `<?if ?>`, `<?ifdef ?>`, or `<?ifndef ?>` tags. They are followed by an xml block, an optional `<?else?>` or `<?elseif ?>` tag, and must end with an `<?endif?>` tag.

Expressions (used in `<?if ?>` and `<?elseif ?>`)

For example: `<?if [expression]?>`

The expression found inside the `<?if ?>` and `<?elseif ?>` tags is a Boolean expression. It adheres to a simple grammar that follows these rules:

The expression is evaluated left to right

Expressions are case-sensitive with the following exceptions:

- Environmental variable names
- These keywords: and, or, not

All variables must use the `$()` syntax or else they will be considered a literal value.

If you want to use a literal `$(`, escape the dollar sign with a second one. For example, `$$(`

Variables can be used to check for existence

Variables can be compared to a literal or another variable

- Comparisons with = and != are string comparisons.
- Comparisons with inequality operators (<, <=, >, >=) must be done on integers.
- If the variable doesn't exist, evaluation will fail and an error will be raised.

The operator precedence is as follows. Note that “and” and “or” have the same precedence:

- ""
- (), \$()
- <, >, <=, >=, =, !=
- Not
- And, Or

Nested parenthesis are allowed.

Literals can be surrounded by quotes, although quotes are not required. Quotes, leading, and trailing white space are stripped off literal values. Invalid expressions will cause an exception to be thrown.

Variables (used in <ifdef ?> and <ifndef ?>)

For example: <?ifdef [variable] ?>

For <ifdef ?>, if the variable has been defined, this statement will be true. <ifndef ?> works in the exact opposite way.

More Examples

Note that these examples will actually each be a no-op because there aren't any tags between the if and endif tags.

```
<?define myValue = "3"?>
<?define system32=$(env.windir)\system32 ?>
<?define B = "good var" ?>
<?define C =3 ?>
<?define IExist ?>

<?if $(var.Iexist) ?><?endif?> <!-- true -->
<?if $(var.myValue) = 6 ?><?endif?> <!-- false -->
<?if $(var.myValue)!=3 ?><?endif?> <!-- false -->
<?if not "x"= "y"?> <?endif?> <!-- true -->
<?if $(env.systemdrive)=a?><?endif?> <!-- false -->
<?if 3 < $(var.myValue)?> <?endif?> <!-- false -->
```

```
<?if $(var.B) = "good VAR"?> <?endif?> <!-- false -->
<?if $(var.A) and not $(env.MyEnvVariable) ?> <?endif?> <!--
<?if $(var.A) Or ($(var.B) And $(var.myValue) >=3)?><?endif?> <!--
<?ifdef IExist ?> <!-- true -->
  <?else?> <!-- false -->
<?endif?>
```

Iteration Statements

There is a single iteration statement, `<?foreach variable-name in semi-colon-delimited-list ?> <?endforeach?>`. When this occurs the preprocessor will

create a private copy of the variable context

set the variable in the foreach statement to an iteration on the semicolon delimited list

generate a fragment with the variable substituted

The affect of this process is that the fragment authored is just template for which the preprocessor will do the iteration to generate the literal of fragments. WiX natively only supports using LCID as a variable name in the `?foreach` statement however, one can use the preprocessor extensions (more below) to provide custom support. The variable name in the `?foreach` statement can be proceeded with a "var.". When the variable is used in the fragment, it must be proceeded with a "var."

An few examples:

```
<?foreach LCID in "1033;1041;1055"?>
  <Fragment Id='Fragment.$(var.LCID)'\>
    <DirectoryRef Id='TARGETDIR'\>
      <Component Id='MyComponent.$(var.LCID)'\ />
    </DirectoryRef>
  </Fragment>
<?endforeach?>
```

or

```
<?define LcidList=1033;1041;1055?>
<?foreach LCID in $(var.LcidList)?>
  <Fragment Id='Fragment.$(var.LCID)'\>
    <DirectoryRef Id='TARGETDIR'\>
      <Component Id='MyComponent.$(var.LCID)'\ />
    </DirectoryRef>
  </Fragment>
<?endforeach?>
```

or

```
filename: ExtentOfLocalization.wxi
<Include>
  <?define LcidList=1033;1041;1055?>
</Include>

and

<?include ExtentOfLocalization.wxi ?>
<?foreach LCID in $(var.LcidList)?>
  <Fragment Id='Fragment.$(var.LCID) '>
    <DirectoryRef Id='TARGETDIR'>
      <Component Id='MyComponent.$(var.LCID)' />
    </DirectoryRef>
  </Fragment>
<?endforeach?>
```

An alternative to the foreach process would be to write the template WiX fragment into a separate file and have another process generate the authoring that will be passed to WiX. The greatest merit of this alternative is that it's easier to debug.

Extensions

WiX has support for preprocessor [extensions](#) via the PreprocessorExtension class. The PreprocessorExtension can provide callbacks with context at foreach initialization, variable evaluation, and the last call before invoking the compiler (for full custom preprocessing). See the `preprocessor.cs` source file and the `preprocessorextension.cs` source file for more information.

Compiler (candle)

The Windows Installer XML compiler is exposed by candle.exe. candle is responsible for preprocessing the input .wxs files into valid well-formed XML documents against the WiX schema, wix.xsd. Then, each post-processed source file is compiled into a .wixobj file.

The compilation process is relatively straight forward. The WiX schema lends itself to a simple recursive descent parser. The compiler processes each element in turn creating new symbols, calculating the necessary references and generating the raw data for the .wixobj file.

The second version of candle is not significantly different from the first implementation. Any changes were either made to enable the new symbol/reference linking or based on feedback from customers. Some of the differences between versions include: the new object file format is XML instead of MSI, modularization of primary keys now happens at link time, and binary streams are imported at link time.

Linker (light)

The Windows Installer XML linker is exposed by light.exe. light is responsible for processing one or more .wixobj files, retrieving metadata from various external files and creating a Windows Installer database (MSI or MSM). When necessary, light will also create cabinets and embed streams in the created Windows Installer database.

The linker begins by searching the set of object files provided on the command line to find the entry section. If more than one entry section is found, light fails with an error. This failure is necessary because the entry section defines what type of Windows Installer database is being created, a MSI (<Product/>) or MSM (</Module/>). It is not possible to create two databases from a single link operation.

While the linker was determining the entry section, the symbols defined in each object file are stored in a symbol table. After the entry section is found, the linker attempts to resolve all of the references in the section by finding symbols in the symbol table. When a symbol is found in a different section, the linker recursively attempts to resolve references in the new section. This process of gathering the sections necessary to resolve all of the references continues until all references are satisfied. If a symbol cannot be found in any of the provided object files, the linker aborts processing with an error indicating the undefined symbol.

After all of the sections have been found, complex and reverse references are processed. This processing is where Components and Merge Modules are hooked to their parent Features or, in the case of Merge Modules, Components are added to the ModuleComponents table. The reverse reference processing adds the appropriate Feature identifier to the necessary fields for elements like, Shortcut, Class, and TypeLib.

Once all of the references are resolved, the linker processes all of the rows retrieving the language, version, and hash for referenced files, calculating the media layout, and including the necessary standard actions to ensure a successful installation sequence. This part of the processing typically ends up generating additional rows that get added associated with the entry section to ensure they are included in the final Windows Installer database.

Finally, light works through the mechanics of generating IDT files and importing them into the Windows Installer database. After the database is fully created, the final post processing is done to merge in any Merge Modules and create a cabinet if necessary. The result is a fully functional Windows Installer database.

Dark

Dark is a tool for converting an MSI or MSM file into a WiX source file. This tool is very useful for getting all your authoring into a WiX source file when you have an existing MSI or MSM. However, you will then need to tweak this file to accomodate different languages and breaking things into fragments.

Visual Studio Package (Votive)

Votive is the code name for the WiX Visual Studio package that allows you to easily create WiX projects, edit WiX files using IntelliSense, and compile/link your project.

Installation

To install the package you must run Votive.msi, which comes bundled with the WiX binaries and gets built as part of the WiX source.

If you have the VSIP SDK installed, you can build the the debug version of Votive.msi, which will install Votive into the Experimental Hive of Visual Studio. This will allow your retail version of Visual Studio to continue to work as before (see the "Developing for Votive" topic for an explanation of the experimental hive).

Developing for Votive

If you want to contribute code to the Votive project or debug Votive, you must download and install the VSIP SDK, available at <http://www.vsipdev.com/downloads/>. You will need the VSIP SDK 2003 and the VSIP SDK 2003 Extras, installed in that order. The SDK is fairly non-invasive and will create an "Experimental Hive" in the registry that will keep your retail version of Visual Studio 2003 unaffected.

To start debugging Votive, set your breakpoints then just hit "F5" in the Wix.sln for Visual Studio. The custom build actions in the Votive project will set up and register Votive on the experimental hive, so running the Votive.msi is not required, nor suggested.

Getting Started

Rob Mensching has written an excellent [MSDN article](#) on using Votive to get started in WiX.

WiX Files

All input files and intermediate files for WiX are xml files. Output is in the form of standard Windows Installer database files.

Source files (.wxs and .wxi) are compiled, producing object files (.wixobj). These objects files are then consumed by the linker which produces Windows Installer database files (.msi or .msm). This is analogous to the C++ model of compiling source code to object files, then linking to produce executables.

Input Files

[Include Files \(.wxs\)](#)

[Source Files \(.wxs\)](#)

Intermediate Files

[Object Files \(.wixobj\)](#)

Output Files

[Installer Database \(.msi\)](#)

[Merge Module \(.msm\)](#)

Include Files (.wxi)

A .wxi file is analogous to .h files for C++. The root element of this file is `<Include/>`. Everything under the root element will be inserted when this file is included in another source or include file.

Source Files (.wxs)

All .wxs files are well-formed XML documents that contain a single root element named `<Wix/>`. The rest of the source file may or may not adhere to the WiX schema before preprocessing. However, after being preprocessed all source files must conform to the WiX schema or they will fail to compile.

The root `<Wix/>` element can contain at most one of the following two elements as children: `<Product/>`, `<Module/>`. However, there can be an unbounded number `<Fragment/>` elements as children of the root `<Wix/>` element. When a source file is compiled into an object file, each instance of these elements creates a new section in the object file. Therefore, these three elements are often referred to as section elements.

It is important to note, that there can be only one `<Product/>` or `<Module/>` section element per source file because they are compiled into special sections called entry sections. Entry sections are used as starting points in the linking process. Sections, entry sections, and the entire linking process are described in greater detail later in this document.

The children of the section elements define the contents of the Windows Installer database. You'll recognize `<Property/>` elements that map to entries in the Property table and a hierarchy of `<Directory/>` elements that build up the Directory table. Most elements contain an "Id" attribute that will act as the primary key for the resulting row in the Windows Installer database. Note, in the first release of the WiX schema the primary key was represented by the text of the element. This location for the primary key was undesirable for several reasons and has been moved to the "Id" attribute. In most cases, the "Id" attribute also defines a symbol when the source file is compiled into an object file.

Object Files (.wixobj)

A .wixobj file is created by the compiler for each source file compiled. The .wixobj file is an XML document that follows the objects.xsd schema defined in the WiX project. As stated above the .wixobj file contains one or more sections that, in turn, contain symbols and references to other symbols.

While the symbols and references are arguably the most important pieces of data in the .wixobj file, they are rarely the bulk of the information. Instead, the majority of most .wixobj files are composed of , and elements that provide the raw data to be placed in the Windows Installer database. In many cases, the linker will not only process the symbols and references but also use and update the raw data from the .wixobj file.

It is interesting to note that the object file schema, objects.xsd, uses camel casing where the source file schema, wix.xsd, uses Pascal casing. This was a conscious choice to indicate that the object files are not intended to be edited by the user. In fact, all schemas that defines data to be processed only by the WiX tools use camel casing.

Windows Installer Installation Packages

The installation package file (.msi) is the basis for vehicle for delivering setup logic to the Windows Installer. For more information on msi files, please see the Windows Installer help file (msi.chm) or visit online at [MSDN](#).

Windows Installer Merge Modules

The merge module file (.msm) is used for sharing setup logic amongst different groups. Basically, a merge module can be created by one group, then merged into another group's installation package. For more information on msi files, please see the Windows Installer help file (msi.chm) or visit online at [MSDN](#).

Building WiX

Simply run "make.bat" in the root of the WiX project. This will build into a release\debug directory by default. Specifying "make.bat ship" will create ship binaries in release\ship. If you installed VS.NET to a non standard directory, specify "make.bat debug fullpathto\devenv.com" to build.

In order to fully build WiX, you must have the following Frameworks and SDKs installed:

NAnt version 0.85 rc3 or higher

.NET Framework 1.1 and SDK

.NET Framework 2.0 (SDK is optional)

PlatformSDK (version 3790.1830 or higher)

- Core SDK
- Web Workshop (IE) SDK
- Internet Information Server (IIS) SDK
- Microsoft Data Access Services (MDAC) SDK
- Microsoft Windows Installer SDK

One of the following Visual Studio 2005 Editions:

- Visual C++ Express Edition
- Professional or higher with Visual C++ installed

HTML Help SDK 1.4 or higher

To build Sconce and Votive, you must have the following SDKs installed:

Visual Studio Partner Integration Program (VSIP) SDK 2003

VSIP SDK 2003 Extras

Both are available at

<http://msdn.microsoft.com/vstudio/partners/default.aspx>

To install Votive on Visual Studio 2005, you must have the Standard edition or higher.

To successfully build the WiX MSBuild tasks, you need to modify NAnt.exe.config file to support the release version of the .NET Framework (instead of the beta 1 release supported in NAnt 0.85 rc3). Make a backup copy of Nant.exe.config and search for the following

<framework> element:

```
<framework
  name="net-2.0"
  ...
```

Replace it with the following element:

```
<framework
  name="net-2.0"
  family="net"
  version="2.0"
  description="Microsoft .NET Framework 2.0"
  runtimeengine=""
  sdkdirectory="${path::combine(sdkInstallRoot, 'bin')}}"
  frameworkdirectory="${path::combine(installRoot, 'v2.0.50727')}}"
  frameworkassemblydirectory="${path::combine(installRoot, 'v2.0.50
  clrversion="2.0.50727"
>
```

Blogs

[Rob Mensching's "when setup isn't just xcopy"](#)

Rob Mensching is the "benevolent dictator" for the Windows Installer XML toolset. He maintains a blog called "[when setup isn't just xcopy](#)". Selected blog entries are reprinted here with his permission.

Rob Mensching's "when setup isn't just xcopy"

Rob Mensching is the "benevolent dictator" for the Windows Installer XML toolset. He maintains a blog called "[when setup isn't just xcopy](#)". Selected blog entries are reprinted here with his permission.

[2004/04/05 - Windows Installer XML \(WiX\) toolset has released as Open Source on SourceForge.net.](#)

[2004/04/14 - So you want to be a Windows Installer XML developer?](#)

[2004/05/11 - Sections, Symbols and References in the Windows Installer XML \(WiX\) toolset.](#)

[2004/05/16 - My philosophical musings about building setup for software.](#)

[2004/05/20 - VBScript \(and Jscript\) MSI CustomActions suck.](#)

[2004/11/22 - Localization and your MSI file.](#)

[2004/11/30 - Creating localized MSI files using WiX toolset and .wxi files.](#)

[Windows Installer XML \(WiX\) toolset has released as Open Source on SourceForge.net](#)

The [Windows Installer Xml \(WiX\) toolset](#) (pronounced “wicks toolset”) appears to have finished propagating around the [SourceForge.net](#) CVS servers, so I can finally start writing this blog entry. As promised in my blog [here](#), [here](#), [here](#), [here](#), and [here](#) the WiX toolset and all of its source code has been released so that you can build Windows Installer databases (MSI and MSM files) the same way most groups inside [Microsoft](#) do. However, a [funny thing happened](#) on the way to the [forum](#). WiX became the first project from Microsoft to be released under an [OSS approved license](#), namely the [Common Public License](#).

Before everyone gets sidetracked by the Open Source implications, let’s talk about exactly what WiX is. WiX is a toolset composed of a compiler, a linker, a lib tool and a decompiler. The compiler, called candle, is used to compile XML source code into object files that contain symbols and references to symbols. The linker, called light, is fed one or more object files and links the references in the object files to the appropriate symbols in other object files. Light is also responsible for collecting all of the binaries, packaging them appropriately, and generating the final MSI or MSM file. The lib tool, called lit, is an optional tool that can be used to combine multiple object files into libraries that can be consumed by light. Finally, the decompiler, called dark, can take existing MSI and MSM files and generate XML source code that represents the package.

So, let me step through a real quick example before sending you off to the [SourceForge project](#) to get the binaries and source code. First, the below is a complete source file that will create a MSI file that installs a test .NET Assembly into the “Program Files\Test Assembly” directory.

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='000C1109-0000-0000-C000-000000000046' Name='TestAss
    <Package Id='000C1109-0000-0000-C000-000000000046' Descriptio

    <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
```

```

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='TestAssemblyProductDirectory' Name='test'
          <Component Id='TestAssemblyProductComponent' Guid='0'
            <File Id='TestAssemblyProductFile' Name='assembly'
              </Component>
            </Directory>
          </Directory>
        </Directory>
      <Feature Id='TestAssemblyProductFeature' Title='Test "ssembly'
        <ComponentRef Id='TestAssemblyProductComponent' />
      </Feature>
    </Product>
  </Wix>

```

Now, to build the MSI file we compile and link the source code like so:

```

E:\wix\examples\test\assembly> candle.exe product.wxs
Microsoft (R) Windows Installer Xml Compiler version 2.0.1510.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

```

```
product.wxs
```

```

E:\wix\examples\test\assembly> light.exe product.wixobj
Microsoft (R) Windows Installer Xml Linker version 2.0.1510.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

```

```
E:\wix\examples\test\assembly> dir
```

```

Volume in drive E is New Volume
Volume Serial Number is 8AC4-6AD2
Directory of E:\wix\examples\test\assembly

```

```

04/05/2004 05:04 <DIR>          .
04/05/2004 05:04 <DIR>          ..
02/23/2004 09:55                891 module.wxs
04/05/2004 05:04            52,736 product.msi
04/05/2004 05:04            4,976 product.wixobj
02/23/2004 09:55            1,281 product.wxs
                4 File(s) 59,884 bytes
                2 Dir(s) 90,014,191,616 bytes free

```

```
E:\wix\examples\test\assembly>
```

I'll discuss more complicated examples in future blog entries and update the documentation (WiX.chm) by distilling any discussions here. While we're on the topic of documentation, let's discuss where WiX is in its

product life-cycle.

First of all, I would say that the WiX toolset is pretty close to Beta2 quality. That means core scenarios (compiling/linking) are very solid, less core scenarios (lib'ing/decompiling) still have some bugs, and the documentation leaves much to be desired. Part of my motivation for pushing the toolset external to Microsoft is to encourage me (and maybe find others) to update the documentation. I'll talk more about that in future blog entries.

That said production quality MSI and MSM files can be produced from the WiX toolset today. Internally, teams such as [Office](#), [SQL Server](#), [BizTalk](#), [Virtual PC](#), [Instant Messenger](#), several [msn.com properties](#), and many others use WiX to build their MSI and MSM files today. When someone encounters a bug, the community tracks the issue down and fixes it. Now, via [SourceForge.net](#), you have an opportunity to be a part of the community as well.

Now, let's talk about why WiX was released as Open Source. First, working on WiX has never been a part of my job description or review goals. I work on the project in my free time. Second, WiX is a very developer oriented project and thus providing source code access increases the pool of available developers. Today, there are five core developers (Robert, K, Reid, and Derek, thank you!) regularly working on WiX in their free time with another ten submitting fixes occasionally. Finally, many parts of the Open Source development process appeal to me. Back in 1999 and 2000, I did not feel that many people inside Microsoft understood what the Open Source community was really about and I wanted to improve that understanding by providing an example.

After four and a half years of part-time development, the WiX design (and most of the code) matured to a point where I was comfortable trying to release it externally. So, last October I started looking for a means to release not only the tools but the source code as well. I thought [GotDotNet](#) was the place. However, at that time, none of the existing [Shared Source licenses](#) were flexible enough to accept contributions from the community. Then, in February, I was introduced to [Stephen Walli](#) who was also working to improve Microsoft's relationship with the Open Source community. Fortunately, Stephen was much farther along than I and had the step-by-step plan how to release an Open Source project from Microsoft using an approved OSS license.

Today, via [WiX on SourceForge](#), you get to see the results of many people's efforts to improve Microsoft from the inside out. I'm not exactly sure what is going to happen next but I'm sure there are quite a few people who are interested to see where this leads. Personally, all I hope is that if you find the WiX toolset useful then you'll join the community and help us improve the toolset.

Copyright © Rob Mensching

[So you want to be a Windows Installer XML developer?](#)

People have started expressing interest in joining the [Windows Installer XML toolset](#) development community so I figured I should get some administrative details out of the way. If you are interested in contributing code to the Windows Installer XML toolset, it is very important to read through all four of these topics.

1) The Windows Installer XML toolset copyright is held by [Microsoft](#).

I want to be very up front about the copyright of the Windows Installer XML toolset and how it affects us as developers. Microsoft is the sponsor of the Windows Installer XML project. Before a contribution can be accepted into the WiX project, the lawyers have asked that we assign our rights to those contributions to Microsoft. By having developers sign a copyright assignment agreement, Microsoft can maintain single legal control of the project. That single legal control enables Microsoft to best defend the project in the future if there was ever any sort of legal challenge.

Before jumping to any conspiracy theories, please note that this copyright assignment is exactly the same process the [Free Software Foundation](#) has you go through if you work on a project they sponsor. Also, in Clause 5 of the Windows Installer XML assignment agreement your rights to your contribution are explicitly granted back to you. If you would like a copy of the assignment agreement, please contact wixadmin@microsoft.com.

2) The Windows Installer XML project is a [benevolent dictatorship](#).

In order to ensure consistency in the schema and maintain the quality of the tools, the Windows Installer XML project's CVS tree is locked down. In other words, commits to the code-base by the general populace are prevented. If you attempt commit changes, CVS will inform you that you have "Insufficient Karma to complete the task."

To have your contribution submitted to the project, please submit an assignment agreement as described above (you only need to do so

once) then send your code diff to WiX-devs@lists.sourceforge.net. The developers there will review the changes and someone will apply them to CVS as quickly as possible.

3) The Windows Installer XML community is a [meritocracy](#).

Those individuals in the community who demonstrate an understanding of the code base by actively participating on the [Windows Installer XML mailing lists](#) and consistently submitting high quality diffs will be given a “Karma boost”. With enough Karma you will earn the ability to commit changes directly to the Windows Installer XML project’s CVS tree.

Commit privileges should not be taken lightly. It is very important that the WiX toolset maintain a high quality bar because many people depend on the tools working properly. Very few developers earn these privileges. In fact, in over four years of development, only five developers have earned commit privileges to the internal Windows Installer XML project.

4) The Windows Installer XML developers are all [volunteers](#).

Everyone (to the best of my knowledge) that works on the Windows Installer XML toolset does so in his or her free time. Please keep that fact in mind when asking for help, submitting code diffs, or interacting with any members of the project. We all want to help to make the Windows Installer XML toolset as solid a tool as possible, but sometimes “real jobs” and “significant others” have to take a higher precedence.

If worse comes to worse, you have access to the source code. Try reading for a while. :)

Copyright © Rob Mensching

Sections, Symbols and References in the Windows Installer XML (WiX) toolset.

Thus far, it seems everyone has been creating one single .wxs source file for their entire MSI or MSM file. This is understandable, since the "Getting Started" topic in the WiX.chm only shows one .wxs file per MSI and MSM. And if you started learning WiX by trying to decompile an existing MSI or MSM, dark will only generate a single .wxs source file for your MSI or MSM file. But the real power of the WiX toolset only becomes apparent when you break up your setup into different *sections* then let the *symbols* and *references* tie your source files back into a cohesive package.

I'll start by showing you the WiX source code then I'll try to explain what it does. Let's assume we have a file called "product.wxs" that looks like this:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='00000000-0000-0000-0000-000000000000' Name='MyProdu
    Version='0.0.0.0' Manufacturer='My Corporation'>
    <Package Description='My Product' Comments='My Product That I
      InstallerVersion='200' Compressed='yes' />

    <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDirectory' Name='MyDir' LongName='My D
      </Directory>
    </Directory>

    <Feature Id='MyFeature' Title='My Product Feature' Level='1'>
      <ComponentRef Id='MyComponent' />
    </Feature>
  </Product>
</Wix>
```

What I've defined above is the skeleton of a MSI product. At the top is the required <Product/> and <Package/> elements that provide the identification information for this package to the Windows Installer. Then I provide the <Media/> element that defines how any file Resources that

are a part of this package should be laid out. In this case, I want all the files compressed into a single cabinet and that cabinet stored as a stream inside the MSI file. Next, I provide my bare bones Directory tree. Finally, this package is finished off with a very simple Feature tree with one Feature containing one Component.

"Hey, wait! Where's the Component definition for 'MyComponent'?" you might ask. Before I can answer that very important question I need to add a couple more examples files. First, let's add another WiX source file called "fragment.wxs" that looks like this:

```
<?xml version="1.0"?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Fragment Id='MyFragment'>
    <DirectoryRef Id='MyDirectory'>
      <Component Id='MyComponent' Guid='00000000-0000-0000-0000-
        <File Id='MyFile' Name='myfile.txt' LongName='My File.t
      </Component>
    </DirectoryRef>
  </Fragment>
</Wix>
```

If we skip the <Fragment/> element the rest of the WiX code should look pretty familiar. I've defined a Component named "MyComponent" (with a bogus GUID) in the "MyDirectory" Directory and noted that any files contained by this Component will be a part of the Media with Disk Id labeled 1. Then I declare that the Component contains a single text file. For good measure, let's say that there is a file called "present.txt" that looks a lot like this:

```
Each day is a gift. That's why we call it the present.
```

Before (finally) explaining in detail how this all works, let's first prove that it works. Here is the output from my compilation and linking.

```
C:\example>candle.exe product.wxs fragment.wxs
Microsoft (R) Windows Installer Xml Compiler version 2.0.1621.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.
product.wxs
fragment.wxs

C:\example>light.exe product.wixobj fragment.wixobj -o product.msi
Microsoft (R) Windows Installer Xml Linker version 2.0.1621.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.
```

```
C:\example>
```

No output from the light means there were no errors so you should now have a "product.msi" file sitting in the same directory with all your other files. You can install that MSI and see it show up in your Add/Remove Programs if you like, but trust me this all works.

"But how did it work?"

Well, when candle compiles your source code it creates an object file (.wixobj) that has zero or more *sections* in it. The elements that are children of the <Wix/> element (namely: <Product/>, <Module/>, and <Fragment/>) define a new section. So in the example above, product.wxs defines one section and fragment.wxs defines another.

Sections contain data and *references*. Most of the data in the section is information that will end up in the final package (MSI file). Some of the data is just information needed by the linker or binder to build the package. For example, the <File/> element shown above contains the necessary information to define a file Resource in the package as well as the "src" attribute that tells the binder where to find the physical file on disk so that the file can be put into a cabinet and inserted into the package. Finally, the data in the section is used to define all of the *symbols* for the section.

A symbol is the unique identifier for a WiX element in your .wxs source file. In general, the symbol for an element maps to the primary key columns of the MSI table the WiX element represents. For example, the <File/> element's "Id" attribute in WiX maps to the MSI File table's File column which is the primary key column. It is pretty safe to assume that all "Id" attributes in the WiX schema represent symbols. If I was to take a stab at the symbols defined in the example source files above, I think this would be the list:

```
product.wxs  
Product:00000000-0000-0000-0000-000000000000  
Media:1  
Directory:TARGETDIR  
Directory:ProgramFilesFolder  
Directory:MyDirectory  
Feature:MyFeature
```

fragment.wxs

```
Fragment:MyFragment  
Componet:MyComponent  
File:MyFile
```

Of course, I might be missing one or two, but hopefully you get the idea of what the compiler thinks is a symbol. If you really want to know for sure, take a look at the tables.xml file for the columns marked "symbol='yes'".

Symbols exist to be referenced. References, the only thing other than data in a section, point at symbols in the current section or other sections. The compiler creates references to symbols when necessary and stores the references at the top of the section in the object file. Obviously elements like <ComponentRef/> or <DirectoryRef/> create references to Components and Directories respectively, but the compiler will create references in other cases as well. For example, the <Component/> element's "DiskId" attribute creates reference to a <Media/> element's "Id" attribute. Since, the .wixobj file contains the references I can easily list them here for you:

product.wixobj

```
<reference table="Component" symbol="MyComponent" />
```

fragment.wixobj

```
<reference table="Directory" symbol="MyDirectory" />  
<reference table="Media" symbol="1" />
```

Note: I have purposely skipped over the complex reference discussion here, but I'll come back to that in some future blog entry.

Thus far, I've only talked about the compiler. Now that we know the basics behind sections, symbols and references we can talk about the details of the linker. This is where the real power of the WiX toolset kicks in. I also believe the linker differentiates the WiX toolset from the other tools I have seen and/or heard of that can build MSI files today.

The linker starts by processing all of the sections in the provided object files looking for an *entry section*. Today there are two types of entry sections: products and modules. As you would expect, when the linker encounters a product entry section it knows it is generating a MSI. If the linker encounters a module entry section the linker knows it is creating a

MSM file. If the linker comes across two entry sections in the object files, it gives up with an error since the linker cannot generate two outputs at the same time. Consider the entry section to be like the "main()" function in a C or C++ program. That's where the linker starts the programs execution.

While the entry section is being located, the linker is also building up the table of symbols from every section from the provided object files. If any symbols are found to be duplicated, the linker will give up with an error. In the C/C++ linker, this error condition is very similar to the case where you define the same variable in the same scope. Once all of the sections have been processed and a single entry section is found, the linker starts resolving references starting at the entry section.

When the current section has a reference that resolves to a symbol in another section the other new section's references are added to the list to be resolved. The process continues until all references are resolved. If a reference cannot be resolved it causes the linker to bail with an error. This error case is similar to the C/C++ linker cannot find a matching function implementation for one of your calls. Also, any sections that are not referenced are ignored.

It is important to note that sections are the atomic unit of linking. In other words, either all of the information in a section is included in your final output or none of it is included. This fact is important to keep in mind when splitting your source code into Fragments. You only need one symbol in a Fragment to be referenced and the entire contents of the Fragment will be a part of your final output.

Before wrapping up this blog entry, let's step through the example we've used so far. Remember, up above, we provided light.fragment.wixobj and the product.wixobj object files to link. The linker would load all of the symbols in those two object files (getting a list much like I described above) and figure out that the section created by the <Product/> element is our entry section.

The linker would then take the only reference in that section (as shown above) and start looking for the symbol "MyComponent" in the Component table. Of course, that reference resolves into our fragment.wixobj. Then the two references from the fragment.wixobj would be resolved. Remember, references from each section must be

resolved. In this case, the "MyDirectory" in the Directory table and "1" in the Media table references are resolved by symbols from the entry section. The linker now happily goes along its merry way finishing the linking process using those two sections to build the final MSI file.

Hopefully this blog entry helps explain some of the inner workings of the WiX toolset so that you can take better advantage of the tools. This write up (or something like it) will be making its way into the WiX documentation so I would appreciate any feedback that makes sections, symbols, and references in the Windows Installer XML toolset make sense.

Copyright © Rob Mensching

[My philosophical musings about building setup for software.](#)

[Mike Gunderloy](#), who has written a [couple articles](#) about the [WiX toolset](#), posted [a comment](#) on a previous [blog entry](#). In the comment, he suggests that splitting a setup project into fragments only moves the problem but doesn't solve it. Thus, he argued that adding a tool to generate the fragments has value. I still didn't agree. Then I realized that I disagreed because I have some philosophies about how to build software and setup for software that I've never posted here. So, I thought I'd share those philosophies today. Please note that these are guidelines that I use when discussing setup build processes with other people not hard rules.

The developer that wrote the feature knows best what needs to be authored into setup.

This part of my philosophy is based on the fact that the person that knows the most about a particular resource is the developer that wrote the resource. Seriously, if the developer who wrote the code doesn't know where his or her files need to be installed or doesn't know what registry keys are necessary to make the feature work (or whatever other resources are necessary) who does?

Yes, there have been (too many) cases in my career where the developer who wrote the code said, "Uhh, I don't know what my code depends on."

However, in those cases it was pretty easy to look at him or her (or his or her manager) and ask, "Well, uhh, shouldn't you?" In every case, they went off and figured out what was necessary to get their software "in the box".

Fundamentally, if developers don't know what their dependencies are there is very little chance their project will have a solid [security](#), [performance](#), or [deployment](#) story. Area experts may be necessary to help developers work through complicated issues but, in general, developers must be aware of what their software is doing.

Setup authoring is a part of the development process.

Every team I have interacted with (the grand majority of Microsoft and

some smaller companies), the developers on the team are expected to add their source files to the makefile before checking in new files to the project. Yet, many of those teams have people who are solely responsible for adding new files to the setup project. In many cases, one set of developers write registry keys into the code for [SelfReg](#) (this is evil, and I'll explain why one day) and the setup developers have to reverse engineer the registry keys out of the built executables to author setup. Why didn't the original developer just author the registry key into setup in the first place?

There was a point in time where it was arguably difficult to distribute the authoring to all developers when you had to buy custom tools that didn't fit well into the development process. However, today there are a few alternatives out there (like the [WiX toolset](#)) that allow setup to be treated like source code. Now there are no excuses I've heard that hold water why setup authoring for the majority of the resources in setup cannot be distributed across the developers in the organization. Doing that distribution leaves only the "look and feel" and integration between the individual pieces of setup to the core setup developers. All other excuses have always been just [whinging](#) (as [Peter](#) might say).

This part of my philosophy and the one above are the reasons I disagree when Mike says that "breaking up the package into multiple source fragments pushes the problem back one level, but doesn't necessarily solve it." Breaking up your setup in to multiple text files enables developers to maintain setup authoring the same way they maintain all the other code that is part of a project. And that brings me to my final point.

Text files should be the only inputs into the build process.

Over the years, developers have created fairly significant processes for tracking, merging, and reverting changes to text files. For example, [Concurrent Version System](#) (better known as CVS) is used heavily by [SourceForge.net](#) can show you the individual lines changed in a text file over time. For a demonstration, take a look at [Compiler.cs from v1.6 to v1.7 in CVS at SourceForge.net for WiX](#). It is not generally possible to visualize the differences between two revisions of a binary file.

Every input into the build process that is manipulated by a human will eventually hose the process at least once. Being able to text search

(using even the simplest tools like [grep](#)) for the exact change that caused the problem significantly improves build throughput. Binary files usually hide the information by requiring custom tools to be opened and queried to find the break.

Of course, once you find the break it is really nice to be able to fix the problem by launching your favorite text editor and tweaking the line(s) of code with the fault. Requiring a tool to be installed on the build machines increases your impedance to fixing the issue.

Ultimately, keeping binary files out of your build process simplifies your life. That fact is why I disagree with Mike when he suggests it makes sense to "put one more tool into the chain, something with a friendly interface that could spit out the source fragments as needed? I could see doing that with Access/Excel, among other things."

Note I do think it is reasonable for developers to use tools that help generate text files. Those text files then can get checked into source control and built as part of the standard process. However, the tools need to generate human friendly text files. Text files that can only be modified with a custom tool are only half a step better than raw binary files.

This is the philosophy that I developed during my tenure in [Office](#) and have promoted across the company for a couple years now. Many teams have had an incredible amount of success with their setup processes when following these guidelines. There have been many cases where I was asked by other teams to talk about the philosophy. At the end of my talk I always left them with, "If you have someone resisting change, have them contact me. I'll be happy to talk to him or her to discuss how successful other teams have been with this process and see if there isn't some way to address their concerns." I'd be happy to do the same for you.

PS: This philosophy has worked out extremely well for me, but if you have a unique situation that you think wouldn't work using these guidelines, I'd be very interested to hear about them. I'm always looking for alternative views (as long as people aren't just whinging <smile/>).

Copyright © Rob Mensching

[VBScript \(and Jscript\) MSI CustomActions suck.](#)

Today was one of those days where you finally get around to looking at the time and wonder where the heck the hours went. It wasn't even like I really got a lot done. I think my [context switch](#) costs have been really high lately. It feels good to finally be home chilling out to the [Perfecto Chills albums](#). I thought I'd relate a short story to you why VBScript (and Jscript for that matter) should not be used for CustomActions in an MSI.

Today, I realized it was 15:39 when a fellow developer, we'll call him Joe, called me at work. My first thought was, "Jeez, it's almost 4 o'clock and I haven't got anything done!" My second thought was, "I bet Joe is screwed." Joe only calls me when the [WiX toolset](#) has completely failed or he has hit the wall with the Windows Installer. Today, Joe had hit the wall.

"Rob, have you been tracking the email thread about the CustomAction of mine that is failing?" I had seen this thread earlier in the day and remembered Joe mentioning something about a VBScript CustomAction. "A little bit, you're not really using VBScript for your CustomAction, are you?"

That was it. Joe was attempting to debug some rather complex issues with a VBScript CustomAction interacting with some COM components during an install. Everything seemed to work fine if he ran the VBScript (slightly modified) in the cscript.exe or wscript.exe hosts. However, when the script executed in the Windows Installers [ActiveScript engine](#) it failed in rather mysterious ways.

Interestingly enough, a Windows Installer developer attached one of the many emails that I send to people when they are having problems using VBScript for CustomActions. In those emails, I always suggest that script never be used for CustomActions in MSI. So, Joe called me and asked, "So what can I do, Rob?"

My answer was simple, "Joe, there is a reason I recommend never using VBScript for CustomActions. It's because there isn't really a whole lot you can do when you get into this kind of situation." Then I provided him

a few ideas that started with attempting to get the script debugger to somehow attach and try to then debug over to the COM component and ended up suggesting getting the command-line debugger attached to the COM component on load. None of which is trivial.

What I don't understand is why people completely disregard dire warnings that certain technologies should not be used in certain circumstances. Yes, I understand it is extremely easy to write CustomActions in VBScript. No, that doesn't make it a good thing to use in your install.

So, I'm blogging here tonight at the end of a very long day to share with you three reasons, I recommend you not use VBScript or Jscript for CustomActions:

- 1. Robust code is difficult write in script.** Setup code must operate on machines that are in an unknown state. In such hostile environments, there are many different ways that code can fail. Properly recovering from error conditions is very important (even if it just results in rollback). "On Error Resume Next" is not conducive to proper error handling in code. For this reason alone, [Microsoft Office](#) banned all script CustomActions from their MSI files. I am admittedly biased but I believe Office has one of the most impressive, smooth, and stable setup programs for the Windows platform, especially considering its complexity.
- 2. Debugging script in the Windows Installer is difficult.** Some might even argue it is impossible to debug script CustomActions. As Joe is going to find out for the next few days, debugging the interactions between the Windows Installer, the scripting engine, and any other objects is a non-trivial task because the tools are so primitive. There are many tools for C/C++ code that can have very low impact on the machine if you are tracking a particularly skittish bug. Maybe I'm only called in when bugs are really hard, but there have been many times I was thankful for [ntsd](#) and [pageheap.exe](#).
- 3. Anti-virus products kill them.** This one just killed me. A couple years after Office banned the use of scripting for CustomActions, [Visual Studio](#) shipped their first MSI setup. They decided it would be okay if there were a few script CustomActions. When customers got the product, [PSS](#) started getting reports of the Visual Studio setup mysteriously failing and rollingback. After some very long calls, PSS

discovered that if the users' anti-virus programs were disabled the installations would succeed. Turns out many of the top name anti-virus programs considered the scripts hosted by the Windows Installer to be virus and would kill the scripts off failing the Visual Studio install.

Anyway, hope you enjoyed the stories and remember, "VBScript and Jscript suck for CustomActions."

Copyright © Rob Mensching

Localization and your MSI file.

Jenny signed off on my [Monday night blog hours](#) so I'm curled up in my big comfy chair with my laptop ready to discuss some details of the Windows Installer. Honestly, every time I sit in this big chair I consider [writing that book](#) again. But not tonight. For tonight we talk about localization and the Windows Installer.

Before I get started, I want to throw out a very important disclaimer up front. I am not a localization expert and I personally have never fully-localized a product. Most of what I'm presenting here is information that I've gleaned from talking to or just watching localizers. The rest of it I stole from the [Windows Installer SDK](#).

For those of you who have not been indoctrinated in building global software, know that "localization" is the process of making your software available for other "locations" (or locales). "Localizers" are the individuals responsible for localizing your software. Obviously most "localizers" speak or read or comprehend more than one culture. This particular talent is one of the major reasons I make a really lousy localizer. I only really understand American English, C/C++, C#, VBScript and a bit of Australian English (from living with [Peter](#) for a few years). But I digress.

Most people think localization is all about translating the text in their program from one language to another. While this is an important part of the process, there are many other facets of localization. For example, directly related to the text translation process but often neglected is the planning for translated text to take more (or less) space in dialog boxes than the first language did. I remember a localizer mentioning to me once that--in general--a dialog box for German text needs to be somewhere around 1.5 times larger than a dialog box with the same text in English. Another important facet of localization is adjusting text and images to be geopolitically appropriate. Words and images accepted in one part of the world are not always appropriate for other parts of the world. Thus it is important to understand the cultures not just the languages when localizing software.

Okay, so that is probably enough to cover the "Localization" part of this blog entry's title, now let's move on "your MSI file". For the remainder of this blog entry, unless I specifically mention it, the term "MSI file" will be

synonymous with "Windows Installer database" (which includes not only MSI files but Merge Modules [.msm files] as well). So what we're really talking about here is localizing your <Products/> and <Modules/> if you use the [WiX toolset](#).

As promised in the beginning, much of what I'm covering here is covered in what I consider the "Windows Installer Bible", the Windows Installer SDK. When I have questions about the way the Window Installer works, I go back and refer to that documentation. Fortunately, for me and this blog, the Windows Installer SDK can get kinda' cryptic at times. So I'm here to add more words to what already exists.

In particular, the [Localization Overview](#) in the Windows Installer SDK is a great place to start (and I expect I will refer to it several times in this blog entry). That help topic does a pretty good job providing a check-list of things to do when localizing your MSI file. I particularly like the first step, plan for localization.

I am fully aware that many people save setup for the end of the product cycle. I personally believe this practice is very reckless and ill-advised (especially consider there are now tools like the WiX toolset that can integrate directly into your build process). However, I've also noticed that localization is often considered after setup! That doubles down the trouble because right when you need to lock-in the [Componentization](#) your product you're going to be adding more files. Bad planning can make this a horrible chore.

So, here's my standard template for success with localization. First, break out all of the localizable text in your product into a separate [resource-only DLL](#). Second, put that resource-only DLL in a sub-directory named for the language of the resource-only DLL. Keep the name of the resource-only DLL the same though. Since I'm an old Office guy, I usually use the [LCID](#) (1033 is American English) for the directory name but I've seen the trend towards using the ISO locale names (en-us is American English) since the Common Language Runtime goes that way. Third, store the default installation's language in your per-application data store. For example, an HKLM registry key is an okay store if your product was installed per-machine and an HKU registry key would be okay if your product was installed per-user. The Registry/@Root="HKMU" value in Windows Installer XML syntax was designed for this type of scenario. Fourth, store the user's current

language preference in a per-user data store (a per-user registry key works okay) when it differs from the installed value. Finally, when you boot your application load the resource-only DLL from the appropriate sub-directory based on the per-user key if it exists and the per-application key if it does not exist.

You can also do more interesting scenarios if you want to take the system's current language into account, but that's for people that know more about localization that I do. Of course, you should also have some fallback plan if your registry keys are all busted. For example, if the per-application registry key was deleted you could repair a portion of your product. I'll try tossing more advanced scenarios in a later blog entry.

Now that you have a plan for your product's organization, you can go back to thinking about your MSI file. First of all, you'll want to think about the [codepage](#) for your MSI file. [Remember](#) MSI files are not Unicode. That means if you pick the wrong codepage for your MSI file that you'll get square boxes or question marks showing up in your database. The Windows Installer SDK talks about [setting the codepage](#) but if you use the WiX toolset you only need to specify your codepage LCID in the Product/@Codepage or Module/@Codepage attribute for your MSI or MSM file respectively. If you're curious, you can see that the WiX toolset does exactly what the Windows installer SDK says in Binder.cs in the Binder's SetDatabaseCodepage() method.

Also note that because the MSI files are not Unicode files they cannot be truly multi-lingual. This is okay because if we skip over the Overview's steps 3 and 4 (I'll discuss those more later), we see in step 5 that the MSI file has a [ProductLanguage Property](#) that must be set to the LCID of the product. That ProductLanguage Property maps to the Product/@Language or Module/@Language attributes. One of the bugs I discussed in my [previous blog entry](#) is enabling the localization of those attributes.

I'm going to lump in the Overview's steps 5, 6, 7, and 8 under the heading "Things That Identify Your Product as a Unique Identity in the World." The world is a scary place out there with lots of other products to collide and otherwise get lost in. Make sure you follow all of these steps to ensure that you can find your product when it comes time to patch or upgrade. I've seen a few cases where setup developers were being particularly lazy and thought they could skip over some of these steps.

They showed up a few months after their products shipped asking how they can target the appropriate localized version of their MSI file.

In one particular case, a product had some geo-political issue preventing the product from being allowed across the border of some country. I don't remember all of the finer details but I believe the final recommendation was to build a new "politically correct" MSI package, send that off for manufacturing and eat the cost of the thousands of CDs that had already been stamped. I can only imagine how much fun they had [creating coasters in their microwave](#) (or how much explaining they had to do with their "higher-ups").

I was originally going to toss the Overview's step 9 in with the above, but (as many of you know) the Component Rules are very near and dear to my heart. Follow my standard template for organizing your product and you'll naturally have to put each resource-only DLL in its own Component. More importantly you will only need to put the non-localized executables in their own Components. If you had localized text in the executables, you'd be in the unfortunate position of needing to create different Components to install the different language versions of the file even though they would all get installed to the same location (a Component Rule violation). Also, as mentioned above, with this product layout I'll be able to demonstrate some interesting advanced install tricks.

Finally, all that is left now is the real localization work (steps 3 and 4 from the Overview). I know this is something of a let down after such a long blog entry, but I'm going to save the details of how the WiX toolset can help with the localization process for later. My hope is to re-finish (er, "un-break") the localization features in the WiX toolset tomorrow night. Then I will try to write the step-by-step process using the WiX toolset using a really simple example. Some time after that, I'll create a more complicated example that takes advantage of some of those advanced tricks I was talking about.

Until next time, take a look at this [localization example](#) from the Windows Installer SDK, and keep coding. You know I am.

Copyright © Rob Mensching

[Creating localized MSI files using WiX toolset and .wxi files.](#)

Tonight we pick up where I left off last week and continue with the topic of localizing your MSI file. If you haven't read [last week's blog entry](#), you should do that now. Yes, it's pretty long. Don't worry I'll wait. There's lots of good background information in there.

Great, I want to cover a couple more things before we really get started. First, just like in my previous blog entry all of the information presented here works equally well for Merge Modules (MSM files) as it does for MSI files. I'll be using an MSI file in my example and I'll use the words "MSI file" a lot (that's how I get such a high page rank for Windows Installer stuff... just kidding) because I'm lazy and get tired of writing MSI/MSM file. Second, I am using the latest build of the [WiX toolset v2.0.2328.0](#) in my examples. This is important because, as you'll note in the [release notes](#), I fixed many localization issues with this release of the toolset. If you want to follow along, be sure you have a recent version of the WiX toolset.

Today there are really two ways to localize your MSI file. [Step 3](#) and [step 4](#) of the [Localization Example](#) in the Windows Installer SDK that I pointed at last week demonstrate those two methods. First, you can export your MSI file's tables to IDT file format, localize that text file then import the IDT file back into your MSI. This method is the fastest way to update information in your MSI file. However, it also requires the most care because you must ensure the codepage of the IDT file matches the codepage of the MSI file or the import will fail with terribly helpful error messages like, "Failed to import your IDT file for some reason. Have a nice day" (note: [::MsiGetLastErrorRecord\(\)](#) will give you more information about the error but it rarely gives you the exact answer to the issue). It is interesting to note that the remarks in [::MsiDatabaseImport\(\) function](#) discourage using this method for updating your MSI file because of the codepage and other IDT encoding issues (like tabs and carriage returns).

The second way to localize data in your MSI file is to use the Windows Installer [SQL Syntax](#) to update the appropriate columns. This method is arguably easier than the previous method because you don't have to

worry about encoding tabs or carriage returns and the APIs will attempt to encode your text the best it can to match the MSI file's current codepage. Unfortunately, this method is also slower than the previous method because the Windows Installer SQL processor is not particularly speedy.

So how about a solution that provides you, the setup developer, with the fastest method to create localized MSI files without needing to worry too much about encoding all of your data in IDT files correctly? What if all you needed to do was to provide the localized data and the codepage for that data (codepage is still necessary because I don't know how to look at several random strings of text and accurately reverse engineer the codepage from them)? What if you could actually compile all of your source code files once then only link the object files together once for each language? How? Well with the [WiX toolset](#), of course.

Admittedly, the WiX toolset's localization features are some of the least documented features in the WiX toolset. In fact, the only documentation is WiX Localization file, .wxi files, schema (wixloc.xsd) and the code in light.cs that processes the .wxi files. So I'm here now to turn that all around with a step-by-step example.

Let's look at a small example source file that installs a simple file with a shortcut. Let's call this source file "example.wxs":

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='????????-????-????-????-????????????' Name='Example
    Language='1033' Version='1.0.0.0' Manufacturer='Microso
  <Package Id='????????-????-????-????-????????????'
    Description='Example Description for Product'
    Comments='Example Product to demonstrate localized D
    InstallerVersion='200' Compressed='yes' />
  <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
  <Directory Id='TARGETDIR' Name='SourceDir'>
    <Directory Id='ProgramFilesFolder' Name='PFiles'>
      <Directory Id='EXAMPLEDIR' Name='example' LongName='Exa
        <Directory Id='LangDir' Name='1033'>
          <Component Id='ExampleComponent' Guid='PUT-GUID-H
            <File Id='ExampleFile' Name='example.txt' src=
              <Shortcut Id='ExampleShortcut'
                Directory='ProgramMenuFolder'
                Name='Example' LongName='Example
                Description='Shortcut to example.
```

```

        </File>
    </Component>
</Directory>
</Directory>
</Directory>
    <Directory Id='ProgramMenuFolder' Name='ProgMenu' />
</Directory>
<Feature Id='ExampleFeature' Title='Example Feature for Produ
    <ComponentRef Id='ExampleComponent' />
</Feature>
</Product>
</Wix>

```

Note, to compile the code above with candle.exe, you'll need to replace "PUT-GUID-HERE" with your own GUID. I don't provide GUIDs in my examples because people like to copy the examples then forget to change the GUID before shipping. Of course, that would be an immediate [Component Rule](#) violation and I don't want to be responsible for that. Also, before we can link that code with light.exe, we'll need to create a text file called, "example.txt". Here's what my example.txt file looks like:

```
Each day is a gift, that's why we call it the present.
```

Okay, after creating example.wxs (and adding your own GUID) and creating example.txt, you should be able to create an "example.msi" file by compiling and linking the files like so:

```

C:\wix>candle example.wxs
Microsoft (R) Windows Installer Xml Compiler version 2.0.2328.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

example.wxs

C:\wix>light example.wixobj
Microsoft (R) Windows Installer Xml Linker version 2.0.2328.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

C:\wix>

```

As always, no news from light.exe is good news. You can install the newly created MSI file using "msiexec /i example.msi" and should notice

a new shortcut in your ProgramMenuFolder ("Start" -> "All Programs" on Windows XP). But I'm sure for you old WiX toolset hacks out there this example is boring. So, let's get to localizing.

If you used the preprocessor, you are already familiar with \$(var.VAR) for defined variables and \$(env.VAR) for environment variables. Localization in the WiX toolset is done by inserting "localization variables". Localization variables look like \$(loc.VAR). Let's look at our modified source file:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2003/01/wi'>
  <Product Id='????????-????-????-????-????????????' Name='Example
    Language='$(loc.LANG)' Version='1.0.0.0' Manufacturer='
  <Package Id='????????-????-????-????-????????????'
    Description='$(loc.Description)'
    Comments='$(loc.Comments)'
    InstallerVersion='200' Compressed='yes' />
  <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
  <Directory Id='TARGETDIR' Name='SourceDir'>
    <Directory Id='ProgramFilesFolder' Name='PFiles'>
      <Directory Id='EXAMPLEDIR' Name='$(loc.ShortDirName)' L
        <Directory Id='LangDir' Name='$(loc.LANG)'>
          <Component Id='ExampleComponent' Guid='PUT-GUID-H
            <File Id='ExampleFile' Name='$(loc.FileName)'
              <Shortcut Id='ExampleShortcut'
                Directory='ProgramMenuFolder'
                Name='Example' LongName='$(loc.Sh
                Description='$(loc.LongShortcutNa
          </File>
        </Component>
      </Directory>
    </Directory>
  </Directory>
  <Directory Id='ProgramMenuFolder' Name='ProgMenu' />
</Directory>
<Feature Id='ExampleFeature' Title='$(loc.FeatureTitle)' Leve
  <ComponentRef Id='ExampleComponent' />
</Feature>
</Product>
</Wix>
```

You should again be able to compile that file but if you try to link you should see error messages such as, "light.exe : fatal error LGHT0023: Localization string 'FeatureTitle' unknown. Ensure that the

\$(loc.FeatureTitle) is defined." That error message basically means we did not provide a Localization file with all of the localizable identifiers and text. So, now we need to create our first .wxl file. I've called mine example1033.wxl and it goes a little like this:

```
<?xml version='1.0'?>
<WixLocalization xmlns='http://schemas.microsoft.com/wix/2003/01/1
  <String Id='LANG'>1033</String>
  <String Id='Description'>Example Description for Product</String>
  <String Id='Comments'>Example Product to demonstrate localized D
  <String Id='ShortDirName'>example</String>
  <String Id='LongDirName'>Example Directory</String>
  <String Id='Filename'>example.txt</String>
  <String Id='ShortShortcutName'>Example</String>
  <String Id='LongShortcutName'>Shortcut to example.txt</String>
  <String Id='FeatureTitle'>Example Feature for Product</String>
</WixLocalization>
```

Now, to get our MSI file back.

```
C:\wix>candle example.wxs
Microsoft (R) Windows Installer Xml Compiler version 2.0.2328.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

example.wxs

C:\wix>light example.wixobj -loc example1033.wxl
Microsoft (R) Windows Installer Xml Linker version 2.0.2328.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

C:\wix>
```

I want to note that (barring any typos) this MSI file should be identical to the first MSI file we created. I also want to note that this will be the last time we compile the example.wxs. Since we have specified all of our localization variables we no longer need to compile to get changes in our MSI file. All we need to do localize our example1033.wxl file into other languages. Since, I don't know any other languages, I'm going to localize our example1033.wxl file into the "Foo language" and use the Japanese LCID, 1041, since I happen to remember that one. Here's the example1041.wxl file localized into the "Foo language":

```
<?xml version='1.0'?>
<WixLocalization xmlns='http://schemas.microsoft.com/wix/2003/01/1
  <String Id='LANGID'>1041</String>
  <String Id='Description'>Foo Foo foo Foo</String>
  <String Id='Comments'>Foo Foo foo foo foo Foo</String>
  <String Id='ShortDirName'>Foo</String>
  <String Id='LongDirName'>Foo Foo</String>
  <String Id='Filename'>foo.txt</String>
  <String Id='ShortShortcutName'>Foo</String>
  <String Id='LongShortcutName'>Foo foo foo.txt</String>
  <String Id='FeatureTitle'>Foo Foo foo Foo</String>
</WixLocalization>
```

Notice how elegant the "Foo language" is. The elegance really is lost in text format. So much of the "Foo language" is transmitted via the pitch and duration of each word. But I digress. Let's build our "Foo language" example.msi file. This will just stomp over our previous example.msi so make sure you uninstall the previous example.msi file using "msiexec /x example.msi" (or you'll have to go to Control Panel -> Add/Remove Programs). Let's link (and only link) our MSI file:

```
C:\wix>light example.wixobj -loc example1041.wxl
Microsoft (R) Windows Installer Xml Linker version 2.0.2328.0
Copyright (C) Microsoft Corporation 2003. All rights reserved.

C:\wix>
```

Now if you install the MSI file you are likely to see square boxes for the ActionText during the progress dialog box. I believe this occurs when you don't have the Japanese fonts necessary to display the Windows Installer's default text messages. In any case, I don't have Japanese fonts installed on my machine so I see square boxes. However, square boxes or no square boxes everything should install just fine. After installing, you too should see a "Foo" shortcut in your ProgramMenuFolder.

That's all there is to .wxl files. Hopefully, you can see how the Localization files can greatly simplify the relationship between you, your localizers, and your setup. I would also like to note that .wxl files are relatively new constructs in the WiX toolset so if you have suggestions how to improve them please feel free to send your feedback to the "wix-

devs at sourceforge.net" mailing list.

And that brings me to my final point. There is one very fatal flaw in the code above. I debated delaying this blog entry to fix the issue but decided the content here was valuable even with the mistake. Have you found it yet? Look closely at the Component/@Guid attribute. Did that value change each time you created a completely different Component like the step 9 in the [Localization Overview](#) suggests? Probably not because you can't currently localize GUID values as described by [this bug on SourceForge](#). However, the value should change because you have very different Shortcuts in the two Components (and the example.txt file is installed to different locations so there is no overlap). So, I apologize profusely for creating an example that violates the [Component Rules](#) and I will fix the bug ASAP.

In the meantime, have fun playing with your .wxi files and keep coding.

Copyright © Rob Mensching

Getting Help

Please see sourceforge.net/projects/wix for more information.

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Wix Schema](#)

uuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}".

Pattern Type

Must match the regular expression: '[(]?[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}[)]?'.

See Also

[Wix Schema](#)

ModularizeType (Simple Type)

Description

None

Enumeration Type

Possible values: {None, Column, Property, Condition, CompanionFile, SemicolonDelimited}

See Also

[Wix Schema](#)

ComponentGuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}", but also allows "PUT-GUID-HERE" for use in examples. It's also possible to have an empty value "".

Pattern Type

Must match the regular expression: '[(]?[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}[)]?|PUT\-\-GUID\-\-HERE|'.

See Also

[Wix Schema](#)

LocalizableInteger (Simple Type)

Description

Values of this type must be an integer or the value can be a localization variable with the format \$(loc.VARIABLE).

Pattern Type

Must match the regular expression: '[0-9][0-9]*|\\$(loc\[_A-Za-z][0-9A-Za-z_]*\)'.

See Also

[Wix Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.extension". The following characters are not allowed: \ ? | > : / * " or less-than. The name must be shorter than 260 characters. The value could also be a localization variable with the format \$(loc.VARIABLE).

Pattern Type

Must match the regular expression: '^[^\\?|><:\^*"]{1,259}|\\$(loc\[[_A-Za-z][0-9A-Za-z_]*\])\$'.

See Also

[Wix Schema](#)

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". The following characters are not allowed: \ ? | > : / * " + , ; = [] less-than, or whitespace. The name cannot be longer than 8 characters and the extension cannot exceed 3 characters. The value could also be a localization variable with the format \$(loc.VARIABLE).

Pattern Type

Must match the regular expression: `'[^\|\?|><:\^*" \+,;=\[\]]{1,8}(\.[^\|\?|><:\^*" \+,;=\[\]]{0,3})?|\$(loc\[A-Za-z][0-9A-Za-z_]*\)'`.

See Also

[Wix Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". The following characters are not allowed: \ | > : / " or less-than. The name must be shorter than 260 characters. The value could also be a localization variable with the format \$(loc.VARIABLE).

Pattern Type

Must match the regular expression: '[^\\|><:/"]{1,259}|\$(loc\[_A-Za-z][0-9A-Za-z_.*\])'.

See Also

[Wix Schema](#)

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

[Wix Schema](#)

HexType (Simple Type)

Description

This type supports any hexadecimal number. Both upper and lower case is acceptable for letters appearing in the number. This type also includes the empty string: "".

Pattern Type

Must match the regular expression: '[0-9A-Fa-f]*'.

See Also

[Wix Schema](#)

autogenuuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". A GUID can be auto-generated by writing all question marks like this: "????????-????-????-????-????????????". Also allows "PUT-GUID-HERE" for use in examples.

Pattern Type

Must match the regular expression: '[(? [0-9A-Fa-f]{8}\-?[0-9A-Fa-f]{4}\-?[0-9A-Fa-f]{4}\-?[0-9A-Fa-f]{4}\-?[0-9A-Fa-f]{12}])|[(? \{8}\-\{4}\-\{4}\-\{4}\-\{12}\)])?|PUT\-\GUID\-\HERE'.

See Also

[Wix Schema](#)

PercentType (Simple Type)

Description

Values of this type are any integers between 0 and 100, inclusive.

xs:nonNegativeInteger Type

- xs:maxInclusive value='100'

See Also

[Wix Schema](#)

VersionType (Simple Type)

Description

Values of this type will look like: "x.x.x.x" where x is an integer from 0 to 65534.

Pattern Type

Must match the regular expression: `'(\d{1,5}\.){3}\d{1,5}'`.

See Also

[Wix Schema](#)

uuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}".

Pattern Type

Must match the regular expression: '[(]?[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}[)]?'.

See Also

[Mmc Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Netfx Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Pubca Schema](#)

uuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF".

Pattern Type

Must match the regular expression: '[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}'.

See Also

[Pubca Schema](#)