Introduction to Windows Installer XML (WiX) toolset

What is WiX?

WiX is a set of tools that allows you to create Windows Installer-based deployment packages for your application. The WiX toolset is based on a declarative XML authoring model. You can use WiX on the command line by using the WiX tools or MSBuild. In addition, there is also a WiX Visual Studio plug-in that supports VS2005, VS2008, and VS2010. The WiX toolset supports building the following types of Windows Installer files:

- Installer (.msi)
- Patches (.msp)
- Merge Modules (.msm)
- Transforms (.mst)

WiX supports a board spectrum of Windows Installer features. In addition, WiX also offers a set of built-in custom actions that can be used and incorporated in Windows Installer packages. The custom actions are offered in a set of WiX extensions. Some common WiX extensions include support for Internet Information System (IIS), Structured Query Language (SQL), the .NET Framework, Visual Studio, and Windows etc.

How does WiX work?

The WiX source code is written in XML format with a .wxs file extension. The WiX tools follow the traditional compile and link model used to create executables from source code. At build time, the WiX source files are validated against the core WiX schema, then processed by a preprocessor, compiler, and linker to create the final result. There are a set of WiX tools that can be used to produce different output types. For a complete list of file types and tools in WiX, see the File Types and the List of Tools sections.

- Using WiX on the Command Line
- Using WiX in Visual Studio
- Using WiX with MSBuild
- Using WiX with NAnt
- How To Guides
- WiX Schema Reference
- Advanced WiX Topics
- Developing for WiX
- Additional Resources

Getting Started

There are several options available to get started learning how to use WiX.

Basic Walkthrough

To get started, you can go to the <u>Creating Installation Packages</u> section which includes a step-by-step walkthrough to create a simple package using WiX.

How To Guides

This help file includes a set of <u>How To Guides</u> that explain how to accomplish common Windows Installer tasks using WiX.

Tutorials

If you prefer to learn from a tutorial, the following options are available:

- <u>Using the WiX Toolset to Integrate Setup into Your Development</u> Process
- http://www.tramontana.co.hu/wix/ Note that this tutorial is currently targeted at WiX 2.0. This tutorial is a great way to ramp up on the WiX toolset if you are new to WiX or are looking for answers to common authoring questions.

If you prefer to learn from audio-visual presentations, the following options are available:

- Blog introduction with video
- Video on Channel 9
- MSDN Radio broadcast

Community

If you prefer to learn by interacting with the community, there is a WiX users mailing list at http://wix.sourceforge.net/mailinglists.html#wix-users.

Integrated Development Environment

If you prefer to learn by using an integrated development environment, there is an overview of WiX editors at http://robmensching.com/blog/archive/2007/11/20/WiX-editors.aspx.

Reverse Engineering

If you prefer to learn by working backward from a Windows Installer package you have already created, you can run the <u>WiX decompiler</u> (Dark) to convert your package into WiX authoring and then recompile it using the <u>WiX compiler</u> (Candle) and <u>WiX linker</u> (Light).

Reading Source Code

If you prefer to learn by reading code, WiX is an open source project, and you can look at the source code by reviewing the How to be a Windows Installer XML Developer topic.

Fix a Bug, Write a Feature

If you prefer to learn by writing code, you can review the following WiX issue trackers:

- Bugs http://sourceforge.net/tracker/? group id=105970&atid=642714
- **Features** http://sourceforge.net/tracker/?group_id=105970&atid=642717

For WiX development assistance, there is a WiX developer mailing list at http://sourceforge.net/mailarchive/forum.php?forum_name=wix-devs.

Getting Help

Please see http://wix.sourceforge.net/ for more information about the WiX toolset. This site includes the following information:

- 1. Links to download weekly releases of the WiX toolset.
- 2. The WiX bug database where you can report new bugs or check the status of existing bugs.
- 3. <u>Mailing lists</u> to ask questions, make suggestions or discuss the WiX toolset with other users and the WiX developers.
- 4. Links to blogs maintained by the WiX developers.

Tools and Concepts

The WiX toolset is tightly coupled with the Windows Installer technology. In order to fully utilize the features in WiX, you must be familiar with the Windows Installer concepts. This section assumes you have a working knowledge of the Windows Installer database format. For information on Windows Installer, see <u>Useful Windows Installer Information</u>.

WiX File Types

There is a set of tools that WiX offers to fulfill the needs of building Windows Installer-based packages. Each tool outputs a type of file that can be consumed as inputs of another tool. After processing through the appropriate tools, the final installer is produced.

To get familiar with the WiX file types, see File Types.

WiX Tools

Once you are familiar with the file types, see how the file types are produced by what WiX tools by visiting <u>List of Tools</u>. For a graphical view of the WiX tools and how they interact with each other, see <u>WiX Toolset Diagram</u>.

WiX Schema

The core WiX schema is a close mirror with the MSI tables. For helpful hints on how the WiX schema maps to MSI tables, see MSI Tables to WiX Schema.

File Types

There are many file types in WiX that are generated from different tools in the toolset. At the highest level, all input files and intermediate files for WiX are XML files. The final output is in the form of standard Windows Installer database files.

For example, to build an MSI or MSP, the compiler processes the source files (.wxs and .wxi) and produces 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.

List of file types

The following list describes the supported file types in WiX:

Extension	Туре	Description
.wxi	Include File	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 inline when this file is included in another source or include file.</include>
.wxl		A .wxl file contains a set of strings used for localizing a product into a specified culture. The root element of this file is <wixlocalization>. The culture is specified by setting the Culture attribute on the <wixlocalization> element.</wixlocalization></wixlocalization>
.wxs		A .wxs file is analogous to a .cpp file for C++. The Root element of this file is <wix>. For more detail, see Additional Information below.</wix>
.wixobj	WiX Object File	A .wixobj file is created by the compiler for each source file compiled. The .wixobj file contains one or more sections that, in turn, contain symbols and references to other symbols. For more detail, see Additional Information below.
.wixout	Output File	A .wixout file is created by the linker which represents the result of linking a set of object files. The .wixout is an XML representation of the final output.
.wixlib	-	A .wixlib file is a library of setup functionality that can be easily shared across different WiX-

		based packages by including it when linking the setup package.
.wixpdb	II -	A .wixpdb file is created by the linker for each final output. It contains the debugging information.
.wixmsp	II .	A .wixmsp file is the XML output generated by linking object files in a patch build.
.wixmst	WiX Transform File	A .wixmst file is an XML representation of the difference between a pair of final outputs or XML outputs.
.msi	Windows Installer Installation Package	An installation package file (.msi) is the basic unit of installation for the Windows Installer.
.msm	Windows Installer Merge Module	A merge module file (.msm) is used to share setup logic across different .msi packages. A merge module can be created by one development team, then merged into another development team's .msi package.
.mst	Windows Installer Transform	A transform file (.mst) is used to apply changes to an .msi file.
.рср	Windows Installer Patch Creation Process	A patch creation properties file (.pcp) is used as an input to the patch building tools provided in the Windows Installer SDK.

Additional Information

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 elements as children: <Product>, <Module>, and <Patch>. 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> or <Patch> 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. 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.

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, most .wixobj files are composed of , <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 define data to be processed only by the WiX tools use camel casing.

List of Tools

To view the usage information of the tools, run /? on the tool via the command line.

Name	Description
	Preprocesses and compiles WiX source files into object files (.wixobj). For more information on compiling, see Compiler . For more information on preprocessing, see Preprocessor .
	Links and binds one or more .wixobj files and creates a Windows Installer database (.msi or .msm). When necessary, Light will also create cabinets and embed streams into the Windows Installer database it creates. For more information on linking, see Linker.
Lit	Combines multiple .wixobj files into libraries that can be consumed by Light.
Dark	Converts a Windows Installer database into a set of WiX source files. This tool is very useful for getting all your authoring into a WiX source file when you have an existing Windows Installer database. However, you will then need to tweak this file to accomodate different languages and breaking things into fragments.
	Generates WiX authoring from various input formats. It is used for harvesting files, Visual Studio projects and Internet Information Server web sites, "harvesting" these files into components and generating Windows Installer XML Source files (.wxs). Heat is good to use when you begin authoring your first Windows Installer package for a product.
Insignia	Inscribes an installer database with information about the digital certificates its external cabs are signed with. For more

	information, see <u>Insignia</u> .		
Melt	Converts an .msm into a component group in a WiX source file.		
	Performs a diff to generate a transform (.wixmst or .mst) for XML outputs (.wixout or .wixpdb) or .msi files.		
Smoke	Runs validation checks on .msi or .msm files.		
	Takes an XML output patch file (.wixmsp) and one or more XML transform files (.wixmst) and produces an .msp file.		
	Enforces standards on WiX source files. WixCop can also be used to assist in converting a set of WiX source files created using an older version of WiX to the latest version of WiX. For more information, see WixCop .		
	Runs validations on a set of XML files and the expected output file. Takes a set of WiX source files and an expected MSI as the input and outputs Pass/Fail.		

Response files

All WiX command-line tools support **response files**, which are text files that contain command-line switches and arguments. Anything you can put on a WiX tool command line can instead go into a response file. Response files are useful when you have command lines that are too long for your command shell. For example, you might want to generate a response file that contains command-line switches and the files that you want to compile with candle.exe:

-nologo -wx 1.wxs 2.wxs 3.wxs

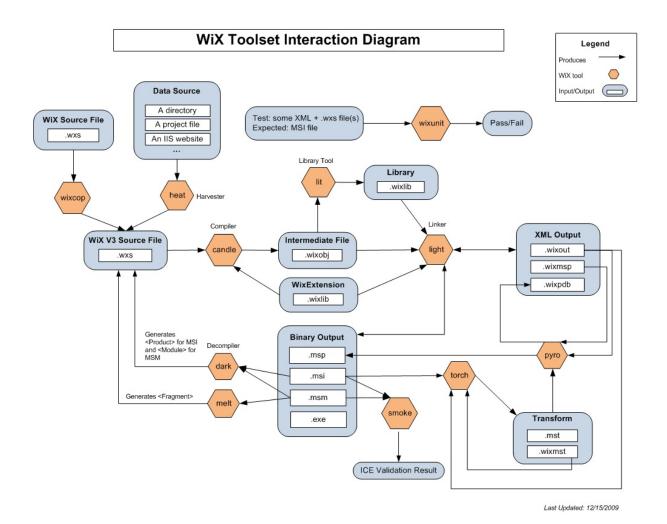
and issue a command like:

candle @listOfFiles.txt

Specify a response file with the @ character, followed immediately by the pathname of the response file, with no whitespace in-between. Response files can appear at the beginning, in the middle, or at the end of command line arguments.

WiX Toolset Diagram

Below is a diagram showing the relationship of all of the WiX tools and the output that they generate.



Preprocessor

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
- PLATFORM the platform (Intel, x64, Intel64) this package is compiled for (set by the Package element's Platform attribute)

NOTE: All built-in directory variables are "\" terminated.

Custom variables <? define ?>

If you want to define custom variables, you can use the <?define?> statement. You can also define variables on the command line using candle.exe using the -d switch. 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 (note: quotes are required if the value or the expansion of other variables in the value contain spaces):

<?define MyVariable = "Hello World" ?>

<?define MyVariable = "\$(var.otherVariableContainingSpaces)" ?>

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 ?>

To define variables on the command line, you can type a command similar to the following:

candle.exe -dMyVariable="Hello World" ...

You can refer to variables in your source that are defined only on the command line, but candle.exe will err when preprocessing your source code if you do not define those variables on the command line.

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
 - The ~= operator is case-insensitive.
- 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 ?>
```

Errors and Warnings

You can use the preprocessor to show meaningful error and warning messages using, <?error error-message ?> and <?warning warning-message?>. When one of these preprocessor instructions is encountered the preprocessor will either display an error and stop the compile or display a warning and continue.

An example:

```
<?ifndef RequiredVariable ?>
    <?error RequiredVariable must be defined ?>
<?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 effect of this process is that the fragment is used as a template by the preprocessor in order to generate a series of fragments. The variable name in the ?foreach statement can be preceded by "var.". When a variable is used inside the text of the fragment, it must be preceded by "var."

An few examples:

or

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.

Escaping

The preprocessor treats the \$ character in a special way if it is followed by a \$ or (. If you want to use a literal \$\$, use \$\$\$\$ instead. Every two \$ characters will be replaced with one. For example, \$\$\$\$ will be replaced with \$\$\$.

Extensions

WiX has support for preprocessor <u>extensions</u> via the PreprocessorExtension class. The PreprocessorExtension can provide callbacks with context at foreach initialization, variable evaluation, function definitions, and the last call before invoking the compiler (for full custom preprocessing).

Compiler

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.

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 or MSM. 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.

Binder Variables

Standard Binder Variables

Some properties are not available until the linker is about to generate, or bind, the final output. These variables are called binder variables and supported binder variables are listed below.

All Versioned Files

The following standard binder variables are available for all versioned binaries.

Variable name	Example usage	Example value
bind.fileLanguage. <i>FileID</i>	!(bind.fileLanguage.MyFile)	1033
bind.fileVersion. <i>FileID</i>	!(bind.fileVersion.MyFile)	1.0.0.0

Assemblies

The following standard binder variables are available for all managed and native assemblies (except where noted), where the File/@Assembly attribute is set to ".net" or "win32".

Variable name	Example usage
bind.assemblyCulture. <i>FileID</i> (managed only)	!(bind.assemblyCulture.MyAs
bind.assemblyFileVersion. <i>FileID</i>	!(bind.assemblyFileVersion.N
bind.assemblyFullName. <i>FileID</i>	!(bind.assemblyName.MyAss

(managed only)	
bind.assemblyName. <i>FileID</i>	!(bind.assemblyName.MyAss
bind.assemblyProcessorArchitecture. <i>FileID</i>	! (bind.assemblyProcessorArc
bind.assemblyPublicKeyToken. <i>FileID</i>	!(bind.assemblyPublicKeyTok
bind.assemblyType. <i>FileID</i> (native only)	!(bind.assemblyType.MyAsse
bind.assemblyVersion. <i>FileID</i>	!(bind.assemblyVersion.MyAs

Localization Variables

Variables can be passed in before binding the output file from a WiX localization file, or .wxl file. This process allows the developer to link one or more .wixobj files together with different .wxl files to produce different localized packages.

Localization variables are in the following format:

!(loc.VariableName)

Custom Binder Variables

You can create your own binder variables using the <u>WixVariable</u> element or by simply typing your own variable name in the following format:

!(bind.VariableName)

Custom binder variables allow you to use the same .wixobj files but specify different values when linking, similar to how localization variables are used. You might use binder variables for different builds, like varying

the target processor architecture.

Insignia

Insignia is a tool used for inscribing an MSI with the digital signatures that its external CABs are signed with.

To sign your external cabs with Insignia, first build your MSI normally, and sign your cabs manually. Then call Insignia with the path to your MSI - Insignia will update your MSI with the digital signature information of its associated external cabs. The file will be updated in-place. Then sign your MSI. This will allow windows installer to verify, at install-time, that the external cabs haven't changed since you built them.

If you use MSBuild, an easier method for doing this exists. In your .wixproj file, set the "InscribeMainOutput" property to "true". Then override the "SignCabs" target, using the "OutputCabList" property as a list of cabs to sign, to sign the external cabs Here's an example signing those cabs using signtool.exe:

```
<Target Name="SignCabs">
<Exec Condition=""%(OutputCabList.FullPath)' != "" Command="Signtool.ex
</Target>
```

Finally, override the "SignMsi" target. Here's a similar example, also using signtool.exe.

```
<Target Name="SignMsi">
  <Exec Condition="'$(OutputMsi)' != "" Command="signtool.exe sign /a &quo
  </Target>
```

This will cause the build process, after linking the MSI, to sign any external cabs, inscribe your MSI with the digital signatures of those cabs, and then sign the MSI, all at the appropriate times during the build process.

WixCop

WixCop serves two main purposes:

- To upgrade WiX authoring to the current schema
- To format WiX authoring according to a set of common formatting

WixCop's command-line syntax is:

```
WixCop.exe [options] sourceFile [sourceFile ...]
```

WixCop takes any number of WiX source files as command-line arguments. Wildcards are permitted. WixCop supports response files containing options and source files, using @responseFile syntax.

WixCop returns the following exit codes:

- 0, when no errors are reported.
- 1, when a fatal error occurs.
- 2, when WixCop violations occur.

WixCop settings files

WixCop supports two settings files. Generally, the primary settings file is your "global" settings and the alternate settings file lets you override the global settings for a particular project.

Settings files are XML with the following structure:

```
<Settings>
<IgnoreErrors>
<Test Id="testId" />
</IgnoreErrors>
<ErrorsAsWarnings>
<Test Id="testId" />
</ErrorsAsWarnings>
<ferorsAsWarnings>
<ExemptFiles>
<File Name="foo.wxs" />
</ExemptFiles>
```

</Settings>

The IgnoreErrors element lists test IDs that should be ignored. The ErrorsAsWarnings element lists test IDs that should be demoted from errors to warnings. The ExemptFiles element lists files that should be skipped. The following table describes the tests that WixCop supports.

WixCop test ID	Description	
Unknown	Internal only: returned when a string cannot be converted to an InspectorTestType.	
InspectorTestTypeUnknown	Internal only: displayed when a string cannot be converted to an InspectorTestType.	
XmlException	Displayed when an XML loading exception has occurred.	
UnauthorizedAccessException	Displayed when a file cannot be accessed; typically when trying to save back a fixed file.	
DeclarationEncodingWrong	Displayed when the encoding attribute in the XML declaration is not 'UTF-8'.	
DeclarationMissing	Displayed when the XML declaration is missing from the source file.	
WhitespacePrecedingCDATAWrong	Displayed when the whitespace preceding a CDATA node is wrong.	
WhitespacePrecedingNodeWrong	Displayed when the whitespace preceding a node is wrong.	
NotEmptyElement	Displayed when an element is not empty as it should be.	
WhitespaceFollowingCDATAWrong	Displayed when the	

	whitespace following a CDATA node is wrong.
WhitespacePrecedingEndElementWrong	Displayed when the whitespace preceding an end element is wrong.
XmlnsMissing	Displayed when the xmlns attribute is missing from the document element.
XmlnsValueWrong	Displayed when the xmlns attribute on the document element is wrong.
CategoryAppDataEmpty	Displayed when a Category element has an empty AppData attribute.
COMRegistrationTyper	Displayed when a Registry element encounters an error while being converted to a strongly-typed WiX COM element.
UpgradeVersionRemoveFeaturesEmpty	Displayed when an UpgradeVersion element has an empty RemoveFeatures attribute.
FeatureFollowParentDeprecated	Displayed when a Feature element contains the deprecated FollowParent attribute.
RadioButtonMissingValue	Displayed when a RadioButton element is missing the Value attribute.
TypeLibDescriptionEmpty	Displayed when a TypeLib element contains a Description element with an empty string value.
ClassRelativePathMustBeAdvertised	Displayed when a RelativePath attribute occurs

	on an unadvertised Class element.
ClassDescriptionEmpty	Displayed when a Class element has an empty Description attribute.
ServiceInstallLocalGroupEmpty	Displayed when a ServiceInstall element has an empty LocalGroup attribute.
ServiceInstallPasswordEmpty	Displayed when a ServiceInstall element has an empty Password attribute.
ShortcutWorkingDirectoryEmpty	Displayed when a Shortcut element has an empty WorkingDirectory attribute.
IniFileValueEmpty	Displayed when a IniFile element has an empty Value attribute.
FileSearchNamesCombined	Displayed when a FileSearch element has a Name attribute that contains both the short and long versions of the file name.
WebApplicationExtensionIdDeprecated	Displayed when a WebApplicationExtension element has a deprecated Id attribute.
WebApplicationExtensionIdEmpty	Displayed when a WebApplicationExtension element has an empty Id attribute.
PropertyValueEmpty	Displayed when a Property element has an empty Value attribute.
ControlCheckBoxValueEmpty	Displayed when a Control element has an empty CheckBoxValue attribute.

RadioGroupDeprecated	Displayed when a deprecated RadioGroup element is found.	
ProgressTextTemplateEmpty	Displayed when a Progress element has an empty TextTemplate attribute.	
RegistrySearchTypeRegistryDeprecated	Displayed when a RegistrySearch element has a Type attribute set to 'registry'.	
WebFilterLoadOrderIncorrect	Displayed when a WebFilter/@LoadOrder attribute has a value that is not more stongly typed.	
SrcIsDeprecated	Displayed when an element contains a deprecated src attribute.	
RequireComponentGuid	Displayed when a Component element is missing the required Guid attribute.	
LongNameDeprecated	Displayed when a an element has a LongName attribute.	
RemoveFileNameRequired	Displayed when a RemoveFile element has no Name or LongName attribute.	
DeprecatedLocalizationVariablePrefix	Displayed when a localization variable begins with the deprecated '\$' character.	
NamespaceChanged	Displayed when the namespace of an element has changed.	
UpgradeVersionPropertyAttributeRequired	Displayed when an UpgradeVersion element is missing the required Property attribute.	

UpgradePropertyChild	Displayed when an Upgrade element contains a deprecated Property child element.		
RegistryElementDeprecated	Displayed when a deprecated Registry element is found.		
PatchSequenceSupersedeTypeChanged	Displayed when a PatchSequence/@Supersede attribute contains a deprecated integer value.		
PatchSequenceTargetDeprecated	Displayed when a deprecated PatchSequence/@Target attribute is found.		
VerbTargetDeprecated	Displayed when a deprecated Verb/@Target attribute is found.		
ProgldlconFormatted	Displayed when a Progld/@Icon attribute value contains a formatted string.		
IgnoreModularizationDeprecated	Displayed when a deprecated IgnoreModularization element is found.		
PackageCompressedIllegal	Displayed when a Package/@Compressed attribute is found under a Module element.		
PackagePlatformsDeprecated	Displayed when a Package/@Platforms attribute is found.		
ModuleGuidDeprecated	Displayed when a deprecated Module/@Guid attribute is found.		
GuidWildcardDeprecated	Displayed when a deprecated guid wildcard value is found.		
FragmentRefIllegal	Displayed when a FragmentRef Element is		

	found.
FileRedundantNames	Displayed when a File/@Name matches a File/@ShortName.

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 <u>CopyFile</u> node nested under a File node. You only need to set the Id, DestinationFolder, and DestinationName attributes.

LaunchCondition Table

This is authored using a <u>Condition</u> node authored under Fragment or Product. You only need to set the Message attribute.

LockPermissions Table

This is authored using <u>Permission</u>.

MoveFile Table

This is authored using a <u>CopyFile</u> 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 <u>Category</u>. 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-	1033	MyComponent	Random	MyFeature
4444-			Data	
55555555555}				

WiX

RemovelniFile

This is authored using <u>IniFile</u>. Just set the Action attribute to 'removeLine' or 'removeTag' as appropriate.

RemoveRegistry Table

This is authored using <u>Registry</u>. Simply set the Action attribute to 'remove' or 'removeKey' (as appropriate) in order to get an entry in the RemoveRegistry table.

Code Pages

Code pages map character codes to actual characters, or graphemes. Code pages are also used to convert from one encoding to another.

Code Pages in Windows Installer

Windows Installer stores strings in a package according to a particular code page. A separate code page is used for the summary information stream and the rest of the package database, which includes the ActionText, Error, Property, and other tables.

For more information about code pages in Windows Installer, read <u>Code Page Handling</u>.

Setting the Code Page using WiX

Top-level elements like <u>Product</u>, <u>Module</u>, <u>Patch</u>, and <u>PatchCreation</u> support a Codepage attribute. You can set this to a valid Windows code page by integer like 1252, or by web name like Windows-1252. UTF-7 and UTF-8 are not officially supported because of user interface issues. Unicode is not supported.

To support authoring a single package that can be localized into multiple languages, you can set the Package (@SummaryCodepage or PatchInformation (@SummaryCodepage element to an localization expression like !(loc.SummaryCodepage). You then define the SummaryCodepage value in a Localization file, typically ending in a .wxl extension. The root WixLocalization element also supports a Codepage attribute that is used to encode the rest of the package database.

You can also set the code page to 0. In this case, Windows Installer treats strings as neutral, meaning that you can only safely use ASCII characters - the first 128 ANSI characters - but the database will be supported across Windows platforms. See <u>Creating a Database with a Neutral Code Page</u> for more information.

For a walkthrough about how to author a build localized packages using WiX see <u>How To: Make your installer localizable</u> and <u>How To: Build a localized version of your installer</u>.

Useful Windows Installer Information

Link to the Windows Installer 4.5 SDK: http://msdn.microsoft.com/en-us/library/aa372866.aspx

List of Windows Installer default properties: http://msdn.microsoft.com/en-us/library/aa370905.aspx

List of Windows Installer operators for conditional expressions: http://msdn.microsoft.com/en-us/library/aa368012.aspx

Creating an Installer Package

In this section, we will cover the basics of creating a simple setup that produces an MSI package using WiX on the command line.

- Creating a Simple Setup
- Creating a Merge Module
- Incorporating a Merge Module
- Adding a Custom Action

Alternatively, WiX can also be used on the command line via MSBuild. For more information, see <u>Using WiX with MSBuild</u>.

Creating a Simple Setup

The goal of this tutorial is to help you to get familiar with WiX and the fundmental building blocks in order to build a simple installable .msi package using WiX. We will start with the basic WiX code to create an .msi package that installs one file.

Step 1: Create the WiX source file

To get started, pick your favorite XML editor (such as Notepad or Visual Studio) 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. Next, add the three lines of text that all .wxs files must contain:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
</Wix>
```

That forms the outer skeleton for our source file. Save the file in the same directory as where the WiX tools are installed (e.g. C:\test).

Step 2: Author the content of the package

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

Replace PUT-GUID-HERE with a GUID value. This will allow you to create an .msi with a ProductCode of "{PUT-GUID-HERE}" with a 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.

Step 3: Adding a file to the package

Now that we have the directory authored, let's put a file in the folder. Create a new text file called readme.txt in the same directory as the product.wxs file and type a message such as "Hello, World!" in the file. Then, we need to modify product.wxs to tell it about the file:

```
<?xml version='1.0'?><Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
 <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'</p>
      Version='1.0.0.0' Manufacturer='Microsoft Corporation' UpgradeCode='I
   <Package Description='My first Windows Installer package'</p>
         Comments='This is my first attempt at creating a Windows Installer da
         Manufacturer='Microsoft Corporation' Installer Version='200' Compres
   <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
   <Directory Id='TARGETDIR' Name='SourceDir'>
     <Directory Id='ProgramFilesFolder' Name='PFiles'>
      <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' Source='readm
        </Component>
      </Directory>
     </Directory>
   </Directory>
   <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
     <ComponentRef Id='MyComponent' />
   </Feature>
 </Product>
</Wix>
```

Step 4: Build the package

Now let's build the package using the following instructions:

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

Step 5: Verify the package

We can now try installing the resulting package and look at the result. Note that msi packages requires administrative privileges in order to install correctly. If will fail silently if you are not installing as an Administrator.

C:\test> msiexec /i product.msi

You can now go to Add/Remove Programs in the Control Panel and see "Test Package" listed there. You can also see that it creates a directory called "Test Program" in your system's "Program Files" folder. The file readme.txt will be installed in the "Test Program" directory. After verifying that installation works as expected, remember to uninstall the .msi so you can rebuild it and install a new version again later.

That's all there is to creating a Windows Installer package. You can do many more advanced things, such as adding setup UI to your .msi, but we've covered the basics. Everything just comes down to filling in the right XML elements.

Creating a Merge Module

Creating a Merge Module is very much like creating a Windows Installer package.

Step 1: Create the WiX source file

Create a new text file called "module.wxs" and put the standard skeleton in it:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
</Wix>
```

Step 2: Author the content of the package

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

Replace PUT-GUID-HERE with a GUID value.

Step 3: Adding a file to the package

Then create a text file called readme2.txt and update the source code to include the new file:

Replace PUT-GUID-HERE with a GUID value.

Step 4: Build the package

Now let's build the package using the following instructions:

C:\test> candle module.wxs

Microsoft (R) Windows Installer Xml Compiler version 1.0.1220.15022 Copyright (C) Microsoft Corporation 2003. All rights reserved

module.wxs

C:\test> light module.wixobj

Microsoft (R) Windows Installer Xml Linker version 1.0.1220.15022 Copyright (C) Microsoft Corporation 2003. All rights reserved

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. See Incorporating a Merge Module

Incorporating a Merge Module into a .wxs File

Merge Modules can only be merged into Windows Installer packages. Once you have a .wxs file that creates a Windows Installer package, it's just a matter of adding two lines (yes, only two lines are necessary) to merge a new Module. In your .wxs source file, add the following lines:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
 <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'</p>
       Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
   <Package Description='My first Windows Installer package'</pre>
        Comments='This is my first attempt at creating a Windows Installer dat
        Manufacturer='Microsoft Corporation' Installer Version='200' Compress
   <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
   <Directory Id='TARGETDIR' Name='SourceDir'>
     <Directory Id='ProgramFilesFolder' Name='PFiles'>
      <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' Source='readme.txt'
        </Component>
        <Merge Id='MyModule' Language='1033' SourceFile='module.msi
      </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.

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

Now when you install the package, you will see that it creates a directory called "Test Program" in your system's "Program Files" folder. The files readme.txt and readme2.txt will be installed in the "Test Program" directory.

Using a Custom Action

WiX provides a set of built-in custom actions that can be used in your installer package. In this example, we will create an installer that creates a shortcut to an URL.

Step 1: Add the extension namespace

To build on top of the simple setup example, let's re-use the same source file and add the following:

```
<?xml version='1.0'?>
<wi>wix xmlns='http://schemas.microsoft.com/wix/2006/wi' xmlns:util='http://scl
 <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'</p>
       Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
   <Package Description='My first Windows Installer package'</p>
        Comments='This is my first attempt at creating a Windows Installer dat
        Manufacturer='Microsoft Corporation' InstallerVersion='200' Compress
   <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
   <Directory Id='TARGETDIR' Name='SourceDir'>
     <Directory Id='ProgramFilesFolder' Name='PFiles'>
       <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' Source='readme.txt'
        </Component>
        <Merge Id='MyModule' Language='1033' SourceFile='module.msm' D
       </Directory>
     </Directory>
   </Directory>
   <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
     <ComponentRef Id='MyComponent' />
     <MergeRef Id='MyModule' />
   </Feature>
 </Product>
</Wix>
```

Step 2: Author the element to create the shortcut

Add the following to the source file::

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi' xmlns:util='http://sche
 <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'</p>
       Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
   <Package Description='My first Windows Installer package'</p>
        Comments='This is my first attempt at creating a Windows Installer dat
        Manufacturer='Microsoft Corporation' InstallerVersion='200' Compress
   <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
   <Directory Id='TARGETDIR' Name='SourceDir'>
     <Directory Id='ProgramFilesFolder' Name='PFiles'>
       <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' Source='readme.txt'
          <util:InternetShortcut Id='Bing' Name='Bing' Target='http://ww
        </Component>
        <Merge Id='MyModule' Language='1033' SourceFile='module.msm' D
       </Directory>
     </Directory>
   </Directory>
   <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
     <ComponentRef Id='MyComponent' />
     <MergeRef Id='MyModule' />
   </Feature>
 </Product>
</Wix>;;;
```

Step 3: Build the installer

To build the installer, you need to pass in the extension that the source file is referencing:

C:\test> candle product.wxs -ext WixUtilExtension

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 -ext WixUtilExtension

Microsoft (R) Windows Installer Xml Linker version 1.0.1220.15022

Copyright (C) Microsoft Corporation 2003. All rights reserved

WixUI Dialog Library

WiX offers a set of built-in Windows Installer-based user interface for installation packages. This section covers the following topics about using the WixUI dialog library:

- Using Built-in WixUI Dialog Sets
- Customizing Built-in WixUI Dialog Sets
- Using Localized Versions of WixUI
- WixUI Dialog Reference

Using Built-in WixUI Dialog Sets

The WixUI dialog library contains the following built-in dialog sets that provide a familiar wizard-style setup user interface.

- 1. WixUI Advanced
- 2. WixUI FeatureTree
- 3. WixUI_InstallDir
- 4. WixUI Minimal
- 5. WixUI_Mondo

The built-in WixUI dialog sets are also customizable, from the bitmaps shown in the UI to adding and removing custom dialogs. See Customizing the WixUI Dialog Sets for additional information.

How to add a built-in WixUI dialog set to a product installer

Assuming you have an existing installer that is functional but is just lacking a user interface, here are the steps you need to follow to include a built-in WixUI dialog set:

1. Add a UIRef element to your setup authoring that has an Id that matches the name of one of the dialog sets described above. For example:

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

2. Pass the -ext and -cultures switches to <u>light.exe</u> to reference the WixUIExtension. For example:

light -ext WixUIExtension -cultures:en-us Product.wixobj -out Product.msi

Note - If you are using WiX in Visual Studio you can add the WixUIExtension using the Add Reference dialog and the necessary command lines will automatically be added when linking your .msi. To do this, use the following steps:

- 1. Open your WiX project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the **WixUIExtension.dll** assembly from the list and click Add
- 4. Close the Add Reference dialog

Customizing Built-in WixUI Dialog Sets

The built-in WixUI dialog sets can be customized in the following ways:

- Specifying a product-specific license agreement file.
- Specifying product-specific setup UI bitmaps.
- Adding an optional checkbox and optional text to the ExitDlg.
- Customizing the text displayed in built-in dialogs.
- Changing the UI sequence of a built-in dialog set.
- Inserting a custom dialog into a built-in dialog set.

Specifying a license file

WixUIExtension.dll includes a default, placeholder license agreement. To specify your product's license, override the default by specifying a WiX variable named WixUILicenseRtf with the value of an RTF file that contains your license text. You can define the variable in your WiX authoring:

```
<WixVariable Id="WixUILicenseRtf" Value="bobpl.rtf" />
```

Alternatively, you can define the variable using the -d switch when running **light**:

light -ext WixUIExtension -cultures:en-us -dWixUILicenseRtf=bobpl.rtf Produc

The file you specify must be in a directory **light** is looking in for files. Use the **-b** switch to add directories.

There is a known issue with the rich text control used to display the text of the license file that can cause the text to appear blank until the user scrolls down in the control. This is typically caused by complex RTF content (such as the RTF generated when saving an RTF file in Microsoft Word). If you run into this behavior in your setup UI, one of the following workarounds will fix it in most cases:

- Open your RTF file in WordPad and save it from there in order to remove the complex RTF content from the file. After saving it, rebuild your MSI.
- Use a dialog set other than the WixUI_Minimal set. This problem typically only occurs when the license agreement screen is the first one displayed during setup, which only happens with the WixUI_Minimal dialog set.

Replacing the default bitmaps

The WixUI dialog library includes default bitmaps for the background of the welcome and completion dialogs and the top banner of the other dialogs. You can replace those bitmaps with your own for product branding purposes. To replace default bitmaps, specify WiX variable values with the file names of your bitmaps, just like when replacing the default license text.

Variable name	Description	Dimensions
WixUIBannerBmp	Top banner	493 × 58
WixUIDialogBmp	Background bitmap used on the welcome and completion dialogs	493 × 312
WixUIExclamationIco	Exclamation icon on the WaitForCostingDlg	32 × 32
WixUIInfolco	Information icon on the cancel and error dialogs	32 × 32
WixUINewIco	Button glyph on the BrowseDlg	16 × 16
WixUIUpIco	Button glyph on the BrowseDlg	16 × 16

Customizing the ExitDlg

The ExitDlg is the <u>dialog in the built-in WixUI dialog sets</u> that is displayed at the end of a successful setup. The ExitDlg supports showing both optional, customizable text and an optional checkbox.

See <u>How To: Run the Installed Application After Setup</u> for an example of how to show a checkbox on the ExitDlg.

To show optional text on the ExitDlg, set the WIXUI_EXITDIALOGOPTIONALTEXT property to the string you want to show. For example:

<Property Id="WIXUI_EXITDIALOGOPTIONALTEXT" Value="Thank you for the content of the content

The optional text has the following behavior:

 The optional text is displayed as literal text, so properties surrounded by square brackets such as [ProductName] will not be resolved. If you need to include property values in the optional text, you must schedule a custom action to set the property. For example:

- Long strings will wrap across multiple lines.
- The optional text is only shown during initial installation, not during maintenance mode or uninstall.

Customizing the text in built-in dialogs

All text displayed in built-in WixUI dialog sets can be overridden with custom strings if desired. In order to do so, you must add a string to your product's WiX localization (.wxl) file that has the same Id value as the string that you want to override. You can find the WixUI string Id values by looking in the file named WixUI_en-us.wxl in the WiX source code.

For example, to override the descriptive text on the WelcomeDlg, you would add the following to a .wxl file in your project:

<String Id="WelcomeDlgDescription">This is a custom welcome message. Clic

Changing the UI sequence of a built-in dialog set

Each of the WixUI dialog sets contains a pre-defined set of dialogs that will be displayed in a specific order. Information about the dialogs included in each built-in WixUI dialog set can be found in the <u>WixUI Dialog Library Reference</u>.

It is possible to change the default sequence of a built-in dialog set. To do so, you must copy the contents of the <Fragment/> that includes the definition of the dialog set that you want to customize from the WiX source code to your project. Then, you must modify the <Publish/> elements to define the exact dialog sequence that you want in your installation experience.

For example, to remove the LicenseAgreementDlg from the <u>WixUI_InstallDir</u> dialog set, you would do the following:

- Copy the full contents of the <Fragment/> defined in WixUI_InstallDir.wxs in the WiX source code to your project.
- 2. Remove the <Publish/> elements that are used to add Back and Next events for the LicenseAgreementDlg.
- Change the <Publish/> element that is used to add a Next event to the WelcomeDlg to go to the InstallDirDlg instead of the LicenseAgreementDlg. For example:

<Publish Dialog="WelcomeDlg" Control="Next" Event="NewDialog" Value of the Control of the Contro

4. Change the <Publish/> element that is used to add a Back event to the InstallDirDlg to go to the WelcomeDlg instead of the LicenseAgreementDlg. For example:

<Publish Dialog="InstallDirDlg" Control="Back" Event="NewDialog" Val</pre>

Inserting a custom dialog into a built-in dialog set

You can add custom dialogs to the UI sequence in a built-in WixUI dialog set. To do so, you must define a <UI/> element for your new dialog. Then, you must copy the contents of the <Fragment/> that includes the definition of the dialog set that you want to customize from the WiX source code to your project. Finally, you must modify the <Publish/> elements to define the exact dialog sequence that you want in your installation experience.

For example, to insert a dialog named SpecialDlg between the WelcomeDlg and the LicenseAgreementDlg in the <u>WixUI_InstallDir</u> dialog set, you would do the following:

- 1. Define the appearance of the SpecialDlg in a <UI/> element in your project.
- 2. Copy the full contents of the <Fragment/> defined in WixUI InstallDir.wxs in the WiX source code to your project.
- 3. Add <Publish/> elements that define the Back and Next events for the SpecialDlg. For example:

```
<Publish Dialog="SpecialDlg" Control="Back" Event="NewDialog" Value
<Publish Dialog="SpecialDlg" Control="Next" Event="NewDialog" Value</pre>
```

4. Change the <Publish/> element that is used to add a Next event to the WelcomeDlg to go to the SpecialDlg instead of the LicenseAgreementDlg. For example:

5. Change the <Publish/> element that is used to add a Back event to the LicenseAgreementDlg to go to the SpecialDlg instead of the WelcomeDlg. For example: <Publish Dialog="LicenseAgreementDlg" Control="Back" Event="NewDi</pre>

Using Localized Versions of WixUI

Using translated UI strings

WixUIExtension includes a set of WiX localization (.wxl) files that contain translated UI text, error and progress text strings for several languages. To specify a UI language for your installer, pass the desired culture value on the command line when calling light. For example:

light -ext WixUIExtension -cultures:fr-fr Product.wixobj -out Product.msi

WixUIExtension includes translated strings for the following languages:

Language name	Culture code	WXL file name
English	en-us	WixUI_en-us.wxl
French	fr-fr	WixUI_fr-fr.wxl
German	de-de	WixUI_de-de.wxl
Italian	it-it	WixUI_it-it.wxl
Japanese	ja-jp	WixUI_ja-jp.wxl
Polish	pl-pl	WixUI_pl-pl.wxl
Russian	ru-ru	WixUI_ru-ru.wxl
Spanish	es-es	WixUI_es-es.wxl

Creating multiple setups with different setup UI languages

You can create a series of .msi files that each use different setup UI languages by calling candle once and then calling light multiple times with different culture values. For example:

candle Product.wxs

light -ext WixUIExtension -cultures:en-us Product.wixobj -out Product_en-us.m light -ext WixUIExtension -cultures:fr-fr Product.wixobj -out Product_fr-fr.msi light -ext WixUIExtension -cultures:it-it Product.wixobj -out Product_it-it.msi light -ext WixUIExtension -cultures:ja-jp Product.wixobj -out Product_ja-jp.msi light -ext WixUIExtension -cultures:pl-pl Product.wixobj -out Product_pl-pl.msi light -ext WixUIExtension -cultures:ru-ru Product.wixobj -out Product_ru-ru.ms light -ext WixUIExtension -cultures:es-es Product.wixobj -out Product_es-es.ms

Using translated error and progress text

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

<UIRef Id="WixUI_ErrorProgressText" />

WixUI Dialog Library Reference

This section explains WixUI dialogs and dialog sets that are included with the WiX toolset.

- WixUI_Advanced Dialog Set
- WixUI_FeatureTree Dialog Set
- WixUI_InstallDir Dialog Set
- WixUI Minimal Dialog Set
- WixUI_Mondo Dialog Set
- WixUI Dialogs

WixUI_Advanced Dialog Set

The WixUI_Advanced dialog set provides the option of a one-click install like WixUI_Minimal, but it also allows directory and feature selection like other dialog sets if the user chooses to configure advanced options.

This dialog set is defined in the file **WixUI_Advanced.wxs** in the WixUIExtension in the WiX source code.

Using WixUI_Advanced

To use WixUI_Advanced, you must include the following information in your setup authoring:

1. A directory with an Id named **APPLICATIONFOLDER**. This directory will be the default installation location for the product. For example:

```
<Directory Id="TARGETDIR" Name="SourceDir">
  <Directory Id="ProgramFilesFolder" Name="PFiles">
       <Directory Id="APPLICATIONFOLDER" Name="My Application Fol ...
       </Directory>
       </Directory>
       </Directory>
```

2. A property with an Id named **ApplicationFolderName** and a value set to a string that represents the default folder name. This property is used to form the default installation location.

For a per-machine installation, the default installation location will be [ProgramFilesFolder][ApplicationFolderName] and the user will be able to change it in the setup UI. For a per-user installation, the default installation location will be [LocalAppDataFolder]Apps\ [ApplicationFolderName] and the user will not be able to change it in the setup UI.

For example:

```
<Property Id="ApplicationFolderName" Value="My Application Folder" /</pre>
```

3. A property with an Id named **WixAppFolder** and a value set to **WixPerMachineFolder** or **WixPerUserFolder**. This property sets the default selected value of the radio button on the install scope

dialog in the setup UI where the user can choose whether to install the product per-machine or per-user. For example:

```
<Property Id="WixAppFolder" Value="WixPerMachineFolder" />
```

It is possible to suppress the install scope dialog in the WixUI_Advanced dialog set so the user will not be able to choose a per-machine or per-user installation. To do this, you must set the **WixUISupportPerMachine** or **WixUISupportPerUser** WiX variables to 0. The default value for each of these variables is 1, and you should not set both of these values to 0 in the same .msi. For example, to remove the install scope dialog and support only a per-machine installation, you can set the following:

```
<WixVariable Id="WixUISupportPerUser" Value="0" />
```

The install scope dialog will automatically set the <u>ALLUSERS</u> property for the installation session based on the user's selection. If you suppress the install scope dialog by setting either of these WiX variable values, you must manually set the ALLUSERS property to an appropriate value based on whether you want a per-machine or per-user installation.

WixUI_Advanced Dialogs

WixUI Advanced includes the following dialogs:

- AdvancedWelcomeEulaDlg
- BrowseDlg
- DiskCostDlg
- FeaturesDlg
- InstallDirDlg
- InstallScopeDlg
- InvalidDirDlg

In addition, WixUI_Advanced includes the following common dialogs that appear in all WixUI dialog sets:

- CancelDlg
- ErrorDlg
- ExitDlg
- FatalError
- FilesInUse
- MaintenanceTypeDlg
- MaintenanceWelcomeDlg
- MsiRMFilesInUse
- OutOfDiskDlg
- OutOfRbDiskDlg
- PrepareDlg
- ProgressDlg
- ResumeDlg
- UserExit
- VerifyReadyDlg
- WaitForCostingDlg

See <u>the WixUI dialog reference</u> for detailed descriptions of each of the above dialogs.

WixUI_FeatureTree Dialog Set

WixUI_FeatureTree is a simpler version of <u>WixUI_Mondo</u> that omits the setup type dialog. Instead, the wizard proceeds 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.

This dialog set is defined in the file **WixUI_FeatureTree.wxs** in the WixUIExtension in the WiX source code.

WixUI_FeatureTree Dialogs

WixUI_FeatureTree includes the following dialogs:

- BrowseDlg
- CustomizeDlg
- DiskCostDlg
- LicenseAgreementDlg
- WelcomeDlg

In addition, WixUI_FeatureTree includes the following common dialogs that appear in all WixUI dialog sets:

- CancelDlg
- ErrorDlg
- ExitDlg
- FatalError
- FilesInUse
- MaintenanceTypeDlg
- MaintenanceWelcomeDlg
- MsiRMFilesInUse
- OutOfDiskDlg
- OutOfRbDiskDlg
- PrepareDlg
- ProgressDlg
- ResumeDlg
- UserExit
- VerifyReadyDlg
- WaitForCostingDlg

See <u>the WixUI dialog reference</u> for detailed descriptions of each of the above dialogs.

WixUI_InstallDir Dialog Set

WixUI_InstallDir does not allow the user to choose what features to install, but it adds a dialog to let the user choose a directory where the product will be installed.

This dialog set is defined in the file **WixUI_InstallDir.wxs** in the WixUIExtension in the WiX source code.

Using WixUI_InstallDir

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. The directory ID must be all uppercase characters because it must be passed from the UI to the execute sequence to take effect. For example:

WixUI_InstallDir Dialogs

WixUI InstallDir includes the following dialogs:

- BrowseDlg
- DiskCostDlg
- InstallDirDlg
- InvalidDirDlg
- LicenseAgreementDlg
- WelcomeDlg

In addition, WixUI_InstallDir includes the following common dialogs that appear in all WixUI dialog sets:

- CancelDlg
- ErrorDlg
- ExitDlg
- FatalError
- FilesInUse
- MaintenanceTypeDlg
- MaintenanceWelcomeDlg
- MsiRMFilesInUse
- OutOfDiskDlg
- OutOfRbDiskDlg
- PrepareDlg
- ProgressDlg
- ResumeDlg
- UserExit
- VerifyReadyDlg
- WaitForCostingDlg

See the WixUI dialog reference for detailed descriptions of each of the above dialogs.

WixUI_Minimal Dialog Set

WixUI_Minimal is the simplest of the built-in WixUI 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 and does not support changing the installation directory.

This dialog set is defined in the file **WixUI_Minimal.wxs** in the WixUIExtension in the WiX source code.

WixUI_Minimal Dialogs

WixUI_Minimal includes the following dialog:

WelcomeEulaDlg

In addition, WixUI_Minimal includes the following common dialogs that appear in all WixUI dialog sets:

- CancelDlg
- ErrorDlg
- ExitDlg
- FatalError
- FilesInUse
- MaintenanceTypeDlg
- MaintenanceWelcomeDlg
- MsiRMFilesInUse
- OutOfDiskDlg
- OutOfRbDiskDlg
- PrepareDlg
- ProgressDlg
- ResumeDlg
- UserExit
- VerifyReadyDlg
- WaitForCostingDlg

See <u>the WixUI dialog reference</u> for detailed descriptions of each of the above dialogs.

WixUI_Mondo Dialog Set

WixUI_Mondo includes a set dialogs that allow granular installation customization options. WixUI_Mondo is appropriate when some product features are not installed by default and there is a meaningful difference between typical and complete installs.

Note: WixUI_Mondo uses <u>SetInstallLevel</u> 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 <u>INSTALLLEVEL Property</u>.

This dialog set is defined in the file **WixUI_Mondo.wxs** in the WixUIExtension in the WiX source code.

WixUI_Mondo Dialogs

WixUI Mondo includes the following dialogs:

- BrowseDlg
- CustomizeDlg
- DiskCostDlg
- LicenseAgreementDlg
- SetupTypeDlg
- WelcomeDlg

In addition, WixUI_Mondo includes the following common dialogs that appear in all WixUI dialog sets:

- CancelDlg
- ErrorDlg
- ExitDlg
- FatalError
- FilesInUse
- MaintenanceTypeDlg
- MaintenanceWelcomeDlg
- MsiRMFilesInUse
- OutOfDiskDlg
- OutOfRbDiskDlg
- PrepareDlg
- ProgressDlg
- ResumeDlg
- UserExit
- VerifyReadyDlg
- WaitForCostingDlg

See the WixUI dialog reference for detailed descriptions of each of the above dialogs.

WixUI Dialogs

The following table describes each of the built-in dialogs that is defined in the WixUI dialog library.

Dialog Name	Description
AdvancedWelcomeEulaDlg	A dialog that displays the end user license agreement. Unlike the LicenseAgreementDlg, it has Advanced and Install buttons instead of Next and Back buttons. This dialog is used by the WixUI_Advanced dialog set to provide the user with a quick way to perform a default installation.
BrowseDlg	A dialog that allows the user to browse for a destination folder.
CancelDlg	A dialog that appears after the user clicks a Cancel button on any dialog and confirms whether or not the user really wants to cancel the installation.
CustomizeDlg	A dialog that displays a feature selection tree with a Browse button, Disk Usage button, and a text box that contains information about the currently selected feature.
DiskCostDlg	A dialog that allows the user to select which drive to install to and that displays disk space usage information for each drive.
	A dialog that displays an error message to the user and can provide an option to retry the previous action.
ExitDlg	A dialog that displays a summary dialog after setup completes successfully. It can also optionally display a checkbox and custom text. For details about how to add

FatalError FeaturesDlg	a checkbox and custom text to this dialog, see Customizing Built-in WixUI Dialog Sets and How To: Run the Installed Application After Setup. A dialog that displays a summary error dialog if setup fails. A dialog that displays a feature selection
	tree with a text box that contains information about the currently selected feature. Unlike the CustomizeDlg, it does not contain Browse or Disk Space buttons.
FilesInUse	A dialog that displays a list of applications that are holding files in use that need to be updated by the current installation process. It includes Retry, Ignore and Exit buttons.
InstallDirDlg	A dialog that has a text box that allows the user to type in a non-default installation path and a Browse button that allows the user to select a non-default installation folder. By default, the InstallDirDlg dialog validates that any path the user enters is valid for Windows Installer: That is, it's a path on a local hard drive, not a network path or on a removable drive. If you wish to disable path validation and allow invalid paths, set the public property WIXUI_DONTVALIDATEPATH to 1.
InstallScopeDlg	A dialog that allows the user to choose to install the product for all users or for the current user.
InvalidDirDlg	A dialog that displays an error if the user selects an invalid installation directory.
LicenseAgreementDlg	A dialog that displays the end user license agreement and includes Back and Next buttons. Unlike the

	AdvancedWelcomeEulaDlg, this dialog does not allow the user to start a default installation.
MaintenanceTypeDlg	A dialog that includes buttons that allow the user to change which features are installed, repair the product or remove the product. It only appears when the user runs setup after a product has been installed.
MaintenanceWelcomeDlg	An introductory dialog that appears when running setup after the product has been installed.
MsiRMFilesInUse	A dialog that is similar to the FilesInUse dialog, but that interacts with Restart Manager. It allows the user to attempt to automatically close applications or ignore the prompt and result in the setup requiring a reboot after it completes.
OutOfDiskDlg	A dialog that informs the user that they have insufficient disk space on the selected drive and advises them to free up additional disk space or reduce the number of features to be installed to the drive.
OutOfRbDiskDlg	A dialog that is similar to the OutOfDiskDlg, but also allows the user to disable Windows Installer rollback functionality in order to conserve disk space required by setup.
PrepareDlg	A simple progress dialog that appears during setup initialization before the first interactive dialog appears.
ProgressDlg	A dialog that appears during installation that displays a progress bar and messages about actions are being performed.

ResumeDlg	An introductory dialog that appears when resuming a suspended setup.
SetupTypeDlg	A dialog that allows the user to choose Typical, Custom or Complete installation configurations.
UserExit	A dialog that that is similar to the FatalError dialog. It displays a summary dialog if the user chooses to cancel setup.
VerifyReadyDlg	A dialog that appears immediately before starting installation. It asks the user for final confirmation before starting to make changes to the system.
WaitForCostingDlg	A dialog that appears if the user advances too far in the setup wizard before Windows Installer has finished calculating disk cost requirements.
WelcomeDlg	An introductory dialog that appears when running setup for a product that has not yet been installed.
WelcomeEulaDlg	A dialog that displays an end user license agreement and allows the user to start installation after accepting the agreement. It is only used by the WixUI_Minimal dialog set and is intended for simple setup programs that do not offer any user configurable options.

Using WiX in Visual Studio

The Visual Studio WiX toolset allows you to easily create WiX projects, edit WiX files using IntelliSense, and compile/link your project within the Visual Studio IDE. The WiX Visual Studio plug-in supports VS2005, VS2008, and VS2010.

For WiX project types, see WiX Project Types.

For WiX item templates, see <u>WiX Item templates</u>.

For the WiX property pages, see WiX Project property pages.

For hints on how to read the default WiX project template, see <u>WiX Project Template</u>.

You can create and build Windows Installer packages using WiX within the Visual Studio IDE. In this section, we will cover the basics of creating a simple setup using Visual Studio and WiX.

- Creating a simple setup
- Using project references and variables

Alternatively, you may also use WiX on the command line by calling the tools directly or using MSBuild. See <u>Creating Installation Packages</u> and <u>Using WiX with MSBuild</u>.

Project Templates

The WiX Visual Studio package provides the following Visual Studio project templates:

- WiX Project used to create a new Windows Installer package
 (.msi) file. Each new WiX project includes a .wxs file that consists of
 a <Product> element that contains a skeleton with the WiX authoring
 required to create a fully functional Windows Installer package. The
 <Package> element includes <Package>, <Media>, <Directory>,
 <Component> and <Feature> elements.
- WiX Library Project used to create a new WiX library (.wixlib) file. A .wixlib file is a library of setup functionality that can be easily shared across different WiX-based packages by including it when linking the setup package. Each new WiX library project includes a .wxs file that consists of an empty <Fragment> element that can be populated with WiX authoring that can be shared by multiple packages.
- WiX Merge Module Project used to create a new Windows Installer merge module (.msm) file. A merge module contains a set of Windows Installer resources that can be shared by multiple Windows Installer installation packages by merging the contents of the module into the .msi package. Each new WiX merge module project includes a .wxs file that consists of a <Module> element that contains a skeleton with the WiX authoring required to create a fully functional merge module. The <Module> element includes <Package>, <Directory> and <Component> elements.

To create a new project:

- 1. Click on File | New | Project on the Visual Studio menu.
- 2. Navigate to the Windows Installer XML node.
- 3. Select the project template and press OK.

Item Templates

WiX Visual Studio package provides the following item templates for WiX projects:

- WiX File a .wxs file pre-populated with the same information as the default WXS file in a WiX Library Project
- WiX Include File a blank .wxi file
- WiX Localization File a blank .wxl file
- Text File a blank .txt file

For more information about WiX file types, please visit the <u>File List</u> section.

To add a new item:

- 1. Right-click on the project node in the Solution Explorer.
- 2. Choose Add | New Item... and select the appropriate item template.
- 3. Type in the item name in the Name field and press Add.

Project Property Pages

To access the WiX project property pages, right-click on a WiX project in the Visual Studio Solution Explorer and choose Properties. WiX projects contain the following property pages:

- Installer
- Build
- Build Events
- Paths
- Tool Settings

Installer Property Page

The Installer tab contains the following configurable options:

- Output name a text box that contains the name of the resultant .msi, .msm or .wixlib file that will be created by the build process.
- Output type a drop-down list that allows you to select the output type (a .msi, .msm or .wixlib file).

Build Property Page

The Build tab contains the following configurable options:

- The **General** section allows you to define configuration-specific constants and specify the culture to build. For more information see Specifying cultures to build.
- The **Messages** section allows you to specify warning levels, toggle treating warnings as errors and verbose output.
- The **Output** section allows you to specify the output path, toggle delete temproary files, suppress output of the wixpdb file, and toggle whether or not to bind files into the library file (if it is a WiX Library project).

Build Events Property Page

The Build Events tab contains the following configurable options:

- **Pre-build event command line** a text box that contains the prebuild events to execute before building the current project.
- Post-build event command line a text box that contains the post-build events to execute after building the current project.
- Run the post-build event a drop-down combo box that allows you
 to specify the conditions in which post-build events should be
 executed.

The Build Events tab contains buttons named **Edit Pre-build...** and **Edit Post-build...** that display edit dialogs for the pre and post-build event command lines. The edit dialogs contain a list of all valid WiX project reference variables and their values based on the current project settings.

Paths Property Page

The Paths tab contains the following configurable options:

- The **Reference Paths** section allows you to define paths you want to use when locating references (WiX extensions and WiX libraries).
- The **Include Paths** section allows you to define paths you want to use when locating WiX Include files.

Tool Settings Property Page

The Tool Settings tab contains the following configurable options:

- The ICE validation section allows you to toggle ICE validation suppression or specify which ICE validation to suppress.
- The **Additional parameters** section allows you to specify command line arguments to pass directly to the WiX tools at build time.

Reading the Default WiX Project Template

Once a WiX project is created, it creates file contains the beginning of the setup code for the project. Everything needed to create an MSI can be added to this file.

Note: If you are not familiar with Windows Installer setup packages, you are strongly encouraged to review the MSDN documentation about the <u>Installation Package</u> before continuing. It will provide a lot of valuable context as we dig into the details of a Windows Installer setup package.

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
 <Product Id="677a7ac3-6e9b-4531-8a61-c31acc301d27" Name="MySetup"</p>
      Version="1.0.0.0" Manufacturer="MySetup" UpgradeCode="ae949a2b
  <Package InstallerVersion="200" Compressed="yes" />
  <Media Id="1" Cabinet="MySetup.cab" EmbedCab="yes" />
  <Directory Id="TARGETDIR" Name="SourceDir">
   <Directory Id="ProgramFilesFolder">
    <Directory Id="INSTALLLOCATION" Name="MySetup">
      <!-- TODO: Remove the comments around this Component element at
      <!-- <Component Id="ProductComponent" Guid="78576f70-ac55-4d9
           <!-- TODO: Insert files, registry keys, and other resources here. --
      <!-- </Component> -->
    </Directory>
   </Directory>
  </Directory>
  <Feature Id="ProductFeature" Title="PUT-FEATURE-TITLE-HERE" Leve</p>
   <ComponentRef Id="ProductComponent" />
  </Feature>
 </Product>
</Wix>
```

If you are familiar with the Windows Installer, the structure of the .wxs file should be familiar. First, the Wix element exists purely to wrap the rest of the content in the file. The Wix element also specifies the namespace, the xmlns attribute that enables validation during compile and autocomplete in Visual Studio via IntelliSense. Next, the Product element defines the required Windows Installer properties used to identify the product, such as the ProductName, <a href="ProductLanguage, and ProductLanguage, and ProductVersion. Third, the Package element contains the attributes for the Summary Information Stream that provides information about the setup package itself. The rest of the elements, except the ComponentRef element, map to Windows Installer tables by the same name, for example the Media table, Directory table, <a href="Component table, and Feature table. The ComponentRef element is used to tie the FeatureComponents table. Components which maps to the entries in the FeatureComponents table.

The default template that is generated when you create a new WiX project will generates a build warning. In the Output window, you may see this warning:

The cabinet 'MySetup.cab' does not contain any files. If this installation contains no files, this warning can likely be safely ignored. Otherwise, please add files to the cabinet or remove it.

Because the WiX project does not yet reference an application, there is nothing to install. Once a file is added to the installer, this warning will go away.

Creating a Simple Setup

In this tutorial, we will create a C# Windows Form Application and then use WiX to create an installer for the application.

Step 1: Create the C# Windows Form Application

- 1. Click **File**, then select **New**, then select **Project**.
- 2. Choose the **Visual C#** node in the **Project Types** tree, then select **Windows Forms Application**.
- 3. Name your application "MyApplication" and press OK.

Step 2: Create the installer for the application

- 1. Click File, then click New, then click Project.
- 2. Choose the **Windows Installer XML** node in the **Project types** tree, then select **WiX Project**
- 3. Name your project "MySetup" and press OK.
- 4. In the **MySetup** project, right-click on the **References** node and choose **Add Reference...**.
- 5. Navigate to the **Projects** tab, click on the **MyApplication** project, and click the **Add** button, and then press OK.
- 6. Build the WiX project.

That's it! Now you have a working installer that installs and uninstalls the application.

Note: After step 5, the appropriate WiX authoring is auto-generated to reference the application project. To disable auto-generation, right-click on the **MyApplication** project reference and go to Properties, then set the Harvest property to False. To manually add the project reference into the WiX source file, open MySetup.wxs and you will see a comment that says:

```
<!-- TODO: Insert your files, registry keys, and other resources here. -->
```

Delete this line and replace it with the following lines of code:

```
<File Id="MyApplicationFile" Name="$(var.MyApplication.TargetFileName)
DiskId="1" KeyPath="yes" />
```

If you type that code into the editor (instead of copying and pasting from this example) you will notice that IntelliSense picks up the valid elements and attributes. IntelliSense with WiX in Visual Studio can save you significant amounts of typing and time when searching for the name of the elements or attributes as you become more comfortable with the WiX language.

That line of code instructs the WiX toolset to add a file resource to the setup package using "MyApplicationFile" as its package identifier. The Name attribute specifies the name for your file when it is installed and the Source attribute specifies where to find the file for packaging during the build. Rather than hard-code values for these attributes into our source code, we use the WiX preprocessor variables that are passed to the WiX compiler. More information about using preprocessor variables, including a table of all supported values, can be found in the <u>Adding Project References topic</u>.

The DiskId attribute instructs the WiX toolset to add this file to the Media element with matching Id attribute. In this example, the MyApplication executable is added to the MySetup.cab cabinet and that cabinet is embedded in the setup package. The KeyPath attribute instructs the WiX toolset to use this file as the key path for the component that contains the file.

Project References and Variables

The WiX project supports adding project references to other projects such as VB and C#. This ensures that build order dependencies are defined correctly within the solution. In addition, it generates a set of WiX preprocessor variables that can be referenced in WiX source files and preprocessor definitions which are passed to the compiler at build time.

To add a project reference to a WiX project:

- 1. Right-click on the **References** node of the project in the Solution Explorer and choose **Add Reference...**.
- 2. In the Add Reference dialog, click on the **Projects** tab.
- 3. Select the desired project(s) and click the **Add** button, and then press OK to dismiss the dialog.

Supported Project Reference Variables

Once a project reference is added, a list of project variables becomes avaliable to be referenced in the WiX source code. Project reference variables are useful when you do not want to have hard-coded values. For example, the \$(var.MyProject.ProjectName) variable will query the correct project name at build time even if I change the name of the referenced project after the reference is added.

The following demonstrates how to use project reference variables in WiX source code:

<File Id="MyExecutable" Name="\$(var.MyProject.TargetFileName)" Source="5</pre>

The WiX project supports the following project reference variables:

Variable name	Example usage	
var. <i>ProjectName</i> .Configuration	\$(var.MyProject.Configuration)	
var. <i>ProjectName</i> .FullConfiguration	\$(var.MyProject.FullConfiguration)	С
var. <i>ProjectName</i> .Platform	\$(var.MyProject.Platform)	Д
var. <i>ProjectName</i> .ProjectDir	\$(var.MyProject.ProjectDir)	C 2
var. <i>ProjectName</i> .ProjectExt	\$(var.MyProject.ProjectExt)	.(
var. <i>ProjectName</i> .ProjectFileName	\$(var.MyProject.ProjectFileName)	Ν
var. <i>ProjectName</i> .ProjectName	\$(var.MyProject.ProjectName)	N
var. <i>ProjectName</i> .ProjectPath	\$(var.MyProject.ProjectPath)	c

		2
var. <i>ProjectName</i> .TargetDir	\$(var.MyProject.TargetDir)	C 2
var. <i>ProjectName</i> .TargetExt	\$(var.MyProject.TargetExt)	.€
var. <i>ProjectName</i> .TargetFileName	\$(var.MyProject.TargetFileName)	N
var. <i>ProjectName</i> .TargetName	\$(var.MyProject.TargetName)	N
var. <i>ProjectName</i> .TargetPath	\$(var.MyProject.TargetPath)	C 2
var. <i>ProjectName.Culture</i> .TargetPath	\$(var.MyProject.en- US.TargetPath)	C 2 L
var.SolutionDir	\$(var.SolutionDir)	C 2
var.SolutionExt	\$(var.SolutionExt)	.5
var.SolutionFileName	\$(var.SolutionFileName)	N
var.SolutionName	\$(var.SolutionName)	N
var.SolutionPath	\$(var.SolutionPath)	C 2

Note: var. *ProjectName. Culture*. TargetPath is only available for projects that have multiple localized outputs (e.g. MSMs).

Using WiX With MSBuild

WiX includes a complete build process (.targets file) for use with MSBuild-based build systems. For more information see the following topics.

- Creating a .wixproj file
- Integrating WiX Projects Into Daily Builds
- Building WiX Projects In Team Foundation Build
- WiX MSBuild Task Reference
- WiX MSBuild Target Reference

Creating a .wixproj File

In order to build WiX using MSBuild, a .wixproj file must be created. The easiest way to create a new .wixproj for your installer is to WiX in Visual Studio because it automatically generates standard msbuild project files that can be built on the command line by simply typing:

msbuild <projectfile>.wixproj

If you do not have Visual Studio available, a .wixproj file can be created using any text editor. The following is a sample .wixproj file that builds an installer consisting of a single product.wxs file. If you want to copy and paste this example, remember to change the <ProjectGuid> value to match your own.

```
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <PropertyGroup>
        <Configuration Condition=" '$(Configuration)' == " ">Debug</Configuration>
        <Platform Condition=" '$(Platform)' == " ">x86</Platform>
        <ProductVersion>3.0</ProductVersion>
        <ProjectGuid>{c523055d-a9d0-4318-ae85-ec934d33204b}
        <SchemaVersion>2.0</SchemaVersion>
        <OutputName>WixProject1</OutputName>
        <OutputType>Package</OutputType>
        <WixTargetsPath Condition=" '$(WixTargetsPath)' == " ">$(MSBuildExtensionsPath)\Microsoft\
    </PropertyGroup>
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|x86' ">
        <OutputPath>bin\$(Configuration)\</OutputPath>
        <IntermediateOutputPath>obj\$(Configuration)\</IntermediateOutputPath>
        <DefineConstants>Debug</DefineConstants>
    </PropertyGroup>
    <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|x86' ">
        <OutputPath>bin\$(Configuration)\</OutputPath>
        <IntermediateOutputPath>obj\$(Configuration)\</IntermediateOutputPath>
    </PropertyGroup>
    <ItemGroup>
        <Compile Include="Product.wxs" />
    </ItemGroup>
    <Import Project="$(WixTargetsPath)" />
</Project>
```

Additional .wxs files can be added using additional <Compile> elements within an ItemGroup. Localization files (.wxl) should be added using the <EmbeddedResource> element within an ItemGroup. Include files (.wxi)

should be added using the <Content> element within an ItemGroup.

Integrating WiX Projects Into Daily Builds

One of the most common reasons for using MSBuild with WiX project files is to integrate the build of an installer into an existing daily build process. This is often coupled with a need to build WiX projects without having to pre-install any WiX tools on the daily build machine. WiX projects and the WiX tools to build them can be added to most daily build processes that support MSBuild using a few simple steps.

Step 1: Check in the WiX Tools

To avoid having to install WiX on build machines you can check all the tools necessary to build WiX projects into your source code control system. Here's how:

- 1. Install WiX on a developer machine using the WiX installer that's appropriate for the machine's architecture (x86 or x64).
- 2. Create a directory in your source code control system to hold the WiX tools. It's common to create a numbered subdirectory matching the version of WiX that you're checking in.
- 3. Copy the contents of **c:\Program Files\Windows Installer XML v3\bin** into the directory created in step 2.
- Copy the contents of c:\Program
 Files\MSBuild\Microsoft\WiX\v3.5 into the directory created in step 2.
- 5. If you use Deployment Tools Foundation or the WiX SDK header files and libraries, create a parallel directory tree to the one you created in step 2 and copy the contents of **c:\Program Files\Windows Installer XML v3\sdk** into that directory.
- 6. Add and check in the files from steps 3 through 5.

The actual file locations in steps 3 through 5 will vary depending on where you installed WiX. On 64-bit operating systems, the files will be located under **c:\Program Files (x86) by default.**

Step 2: Modify Your .wixproj File

After checking the WiX tools into source code control the .wixproj file must be modified to point to the location of the checked in tools. Open the .wixproj file in any text editor, such as Visual Studio, and add the following to the file anywhere between the <Project> element before the <Import> element:

The WixToolPath must be set to point to the location of the WiX tools directory created in Step 1. The method used to reference the location will vary depending on your build system, but common choices are an MSBuild property that is set via an environment variable (such as **\$(BinariesRoot)** in a Team Foundation Server build) or a custom property passed in on the command-line.

You can also use a relative path to the directory (such as ..l..ltoolsl), but note that the WixTargetsPath property value must be relative to the .wixproj project file that uses it. The WixTasksPath property is used inside wix.targets to load WixTasks.dll; its value, if a relative path, must be relative to the wix.targets file. Those two files usually live together, so the value would be WixTasks.dll with no extra path information.

Building WiX Projects In Team Foundation Build

Once you have created a <u>WiX project file</u>, you need to perform some additional steps in order to successfully build the WiX project in Team Foundation Build. Without these additional steps, the WiX project will be ignored by default by Team Foundation Build even though it is an MSBuild-compatible project.

Step 1: Update the Solution Build Configuration

By default, WiX projects will not be built when building the 'Any CPU' platform because Windows Installer packages are CPU-specific. As a result, you need to use the following steps to update the solution build configuration to include your WiX project and its dependencies as part of a Team Foundation Build.

- 1. In the solution, open Configuration Manager (Build | Configuration Manager).
- 2. Set the 'Debug' configuration as the active configuration.
- 3. Select the 'x86' platform that you plan to build from the drop-down list.
- 4. Ensure that the WiX project is checked in the 'Build' column.
- 5. Ensure that any project references that the WiX project uses are also checked in the 'Build' column.
- 6. Set the 'Release' configuration as the active configuration.
- 7. Repeat steps 3-5 to ensure that the WiX project and its dependencies will build for the 'Release' configuration.
- 8. If you plan to build the 'x64' platform, repeat steps 3-7 for the 'x64' platform.
- 9. Close Configuration Manager and save the solution.

Step 2: Add the Build Configurations to TFSBuild.proj

Now that you have added the WiX project and its dependent projects to the 'x86' and/or 'x64' build configurations, Team Foundation Build will build your WiX project in these build configurations. However, these build configurations may not be specified in your Team Foundation Build Definition (TFSBuild.proj).

When you create a new Build Definition, you can select the 'Debug/Mixed Platforms' and 'Release/Mixed Platforms' build configurations to build all projects in your solution, including WiX projects.

If you have an existing Build Definition, you need to use the following steps to modify it so it will build WiX projects along with the other projects in your solution.

- 1. Right-click on the Build Definition and select View Configuration Folder.
- 2. Check out and open the file named TFSBuild.proj.
- Add the following build configurations to the <ConfigurationToBuild> section if they do not already exist there, or update them if they do already exist:

4. Close, save and check in the changes to TFSBuild.proj.

After making the above changes and queuing the build, you will see folders named 'Debug' and 'Release' in the build output. Each of these folders will contain a sub-folder named 'en-us' (or another culture

depending on the settings in the WiX project) that contains the built Windows Installer package.

WiX MSBuild Task Reference

This section explains MSBuild tasks that are included with the WiX toolset.

- Candle Task
- HeatDirectory Task
- HeatFile Task
- HeatProject Task
- Light Task
- Lit Task

Candle Task

The Candle task wraps <u>candle.exe</u>, the WiX compiler. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the Candle task:

The following table describes the common WiX MSBuild parameters that are applicable to the **Candle** task.

Parameter	Description
SuppressAllWarnings	Optional boolean parameter.
	Specifies that all warnings should be suppressed. This is equivalent to the -sw switch.
SuppressSchemaValidation	Optional boolean parameter.
	Specifies that schema validation of documents should be suppressed. This is equivalent to the -ss switch.
SuppressSpecificWarnings	Optional string parameter.
	Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.
TreatSpecificWarningsAsErrors	Optional string parameter.

	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.
TreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all warnings should be treated as errors. This is equivalent to the -wx switch.
VerboseOutput	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the **Candle** task.

Parameter	Description
CompilerAdditionalOptions	Optional string parameter.
	Specifies additional command line parameters to append when calling candle.exe.
CompilerSuppressAllWarnings	Optional boolean parameter.
	Specifies that all compiler warnings should be suppressed. This is equivalent to the -sw switch in candle.exe.
CompilerSuppressSchemaValidation	Optional boolean parameter.
	Specifies that the compiler should suppress schema

	validation of documents. This is equivalent to the -ss switch in candle.exe.
CompilerSuppressSpecificWarnings	Optional string parameter.
	Specifies that certain compiler warnings should be suppressed. This is equivalent to the -sw[N] switch in candle.exe.
CompilerTreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certain compiler warnings should be treated as errors. This is equivalent to the -wx[N] switch in candle.exe.
CompilerTreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all compiler warnings should be treated as errors. This is equivalent to the -wx switch in candle.exe.
CompilerVerboseOutput	Optional boolean parameter.
	Specifies that the compiler should provide verbose output. This is equivalent to the -v switch in candle.exe.
DefineConstants	Optional string parameter.
	Specifies a semicolon- delimited list of preprocessor variables.

	This is equivalent to the - d <name>[=<value>] switch in candle.exe.</value></name>
SuppressFilesVitalByDefault	Optional boolean parameter.
	Specifies that the compiler should suppress marking files as vital by default. This is equivalent to the -sfdvital switch in candle.exe.
PreprocessToStdOut	Optional boolean parameter.
	Specifies that the compiler should output preprocessing information to stdout. This is equivalent to the -p switch in candle.exe.
PreprocessToFile	Optional string parameter.
	Specifies that the compiler should output preprocessing information to a file. This is equivalent to the -p <file> switch in candle.exe.</file>
IncludeSearchPaths	Optional string parameter.
	Specifies directories to add to the compiler include search path. This is equivalent to the -I <dir>switch in candle.exe.</dir>
InstallerPlatform	Optional string parameter.

	Specifies the processor architecture for the package. Valid values are x86, x64, and ia64. (Deprecated values include intel for x86 and intel64 for ia64.) This is equivalent to the -arch switch in candle.exe.
	Sets the sys.BUILDARCH preprocessor variable and, when the value is x64 or ia64, defaults the Win64 attribute to "yes" on all Package, Component, CustomAction, and RegistrySearch elements in the source file.
OnlyValidateDocuments	Optional boolean parameter.
	Specifies that the compiler should only validate documents. This is equivalent to the -zs switch in candle.exe.
Pedantic	Optional boolean parameter.
	Specifies that the compiler should display pedantic messages. This is equivalent to the -pedantic switch in candle.exe.
ShowSourceTrace	Optional boolean parameter.

Specifies that the compiler should show source trace information for errors, warnings and verbose messages. This is equivalent to the -trace switch in candle.exe.

HeatDirectory Task

The **HeatDirectory** task wraps heat.exe, the WiX harvester, using the **dir** harvesting type. Authoring is generated for type libraries and self-registration from DllRegisterServer for any files found in directories. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the **HeatDirectory** task:

```
< Heat Directory
NoLogo="$(HarvestDirectoryNoLogo)"
SuppressAllWarnings="$(HarvestDirectorySuppressAllWarnings)"
SuppressSpecificWarnings="$(HarvestDirectorySuppressSpecificWarnings)"
ToolPath="$(WixToolPath)"
TreatWarningsAsErrors="$(HarvestDirectoryTreatWarningsAsErrors)"
TreatSpecificWarningsAsErrors="$(HarvestDirectoryTreatSpecificWarningsAs
VerboseOutput="$(HarvestDirectoryVerboseOutput)"
AutogenerateGuids="$(HarvestDirectoryAutogenerateGuids)"
GenerateGuidsNow="$(HarvestDirectoryGenerateGuidsNow)"
OutputFile="$(IntermediateOutputPath) %(HarvestDirectory.Filename) dir.wx
SuppressFragments="$(HarvestDirectorySuppressFragments)"
SuppressUniqueIds="$(HarvestDirectorySuppressUniqueIds)"
Transforms="%(HarvestDirectory.Transforms)"
Directory="@(HarvestDirectory)"
ComponentGroupName="%(HarvestDirectory.ComponentGroupName)"
DirectoryRefId="%(HarvestDirectory.DirectoryRefId)"
KeepEmptyDirectories="%(HarvestDirectory.KeepEmptyDirectories)"
PreprocessorVariable="%(HarvestDirectory.PreprocessorVariable)"
SuppressCom="%(HarvestDirectory.SuppressCom)"
SuppressRootDirectory="%(HarvestDirectory.SuppressRootDirectory)"
SuppressRegistry="%(HarvestDirectory.SuppressRegistry)" />
```

The following table describes the common WiX MSBuild parameters that are applicable to the **HeatDirectory** task.

Parameter	Description
NoLogo	Optional boolean parameter.
	Specifies that the tool logo should be suppressed. The default is false . This is equivalent to the -nologo switch.
SuppressAllWarnings	Optional boolean parameter.
	Specifies that all warnings should be suppressed. The default is false . This is equivalent to the -sw switch.
SuppressSpecificWarnings	Optional string parameter.
	Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.
TreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.
TreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all warnings should be treated as errors. The default is false . This is equivalent to the -wx switch.
VerboseOutput	Optional boolean parameter.
	Specifies that the tool should provide verbose output. The default is false . This is equivalent to the -v switch.

The following table describes the parameters that are common to all heat tasks that are applicable to the **HeatDirectory** task.

Parameter	Description
AutogenerateGuids	Optional boolean parameter.
	Whether to generate authoring that relies on auto-generation of component GUIDs. The default is \$(HarvestAutogenerateGuids) if specified; otherwise, true .
GenerateGuidsNow	Optional boolean parameter.
	Whether to generate authoring that generates durable GUIDs when harvesting. The default is \$(HarvestGenerateGuidsNow) if specified; otherwise, false .
OutputFile	Required item parameter.
	Specifies the output file that contains the generated authoring.
SuppressFragments	Optional boolean parameter.
	Whether to suppress generation of separate fragments when harvesting. The default is \$(HarvestSuppressFragments) if specified; otherwise, true .
SuppressUniquelds	Optional boolean parameter.
	Whether to suppress generation of unique component IDs. The default is \$(HarvestSuppressUniqueIds) if specified; otherwise, false .
Transforms	Optional string parameter.
	XSL transforms to apply to all generated WiX authoring. Separate multiple transforms with semicolons. The default is \$(HarvestTransforms) if specified.

The following table describes the parameters that are specific to the $\mbox{\it HeatDirectory}$ task.

Parameter	Description
Directory	Required item group parameter.
	The list of directories to harvest.
ComponentGroupName	Optional string parameter.
	The name of the ComponentGroup to create for all the generated authoring.
DirectoryRefld	Optional string parameter.
	The ID of the directory to reference instead of TARGETDIR.
KeepEmptyDirectories	Optional boolean parameter.
	Whether to create Directory entries for empty directories.
PreprocessorVariable	Optional string parameter.
	Substitute SourceDir for another variable name (ex: var.Dir).
SuppressCom	Optional boolean parameter.
	Suppress generation of COM registry elements. The default is false .
SuppressRegistry	Optional boolean parameter.
	Suppress generation of a Directory element for the parent directory of the file. The default is false .
SuppressRootDirectory	Optional boolean parameter.
	Suppress generation of any registry elements. The default is false .

HeatFile Task

The **HeatFile** task wraps <u>heat.exe</u>, the WiX harvester, using the **file** harvesting type. Authoring is generated for type libraries and self-registration from DllRegisterServer. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the **HeatFile** task:

```
<HeatFile
NoLogo="$(HarvestFileNoLogo)"
SuppressAllWarnings="$(HarvestFileSuppressAllWarnings)"
SuppressSpecificWarnings="$(HarvestFileSuppressSpecificWarnings)"
ToolPath="$(WixToolPath)"
TreatWarningsAsErrors="$(HarvestFileTreatWarningsAsErrors)"
TreatSpecificWarningsAsErrors="$(HarvestFileTreatSpecificWarningsAsError
VerboseOutput="$(HarvestFileVerboseOutput)"
AutogenerateGuids="$(HarvestFileAutogenerateGuids)"
GenerateGuidsNow="$(HarvestFileGenerateGuidsNow)"
OutputFile="$(IntermediateOutputPath) %(HarvestFile.Filename) file.wxs"
SuppressFragments="$(HarvestFileSuppressFragments)"
SuppressUniqueIds="$(HarvestFileSuppressUniqueIds)"
Transforms="%(HarvestFile.Transforms)"
File="@(HarvestFile)"
ComponentGroupName="%(HarvestFile.ComponentGroupName)"
DirectoryRefId="%(HarvestFile.DirectoryRefId)"
PreprocessorVariable="%(HarvestFile.PreprocessorVariable)"
SuppressCom="%(HarvestFile.SuppressCom)"
SuppressRegistry="%(HarvestFile.SuppressRegistry)"
SuppressRootDirectory="%(HarvestFile.SuppressRootDirectory)" />
```

The following table describes the common WiX MSBuild parameters that are applicable to the **HeatFile** task.

Parameter	Description

NoLogo	Optional boolean parameter.
	Specifies that the tool logo should be suppressed. The default is false . This is equivalent to the -nologo switch.
SuppressAllWarnings	Optional boolean parameter.
	Specifies that all warnings should be suppressed. The default is false . This is equivalent to the -sw switch.
SuppressSpecificWarnings	Optional string parameter.
	Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.
TreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.
TreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all warnings should be treated as errors. The default is false . This is equivalent to the -wx switch.
VerboseOutput	Optional boolean parameter.
	Specifies that the tool should provide verbose output. The default is false . This is equivalent to the -v switch.

The following table describes the parameters that are common to all heat tasks that are applicable to the **HeatFile** task.

Parameter	Description
AutogenerateGuids	Optional boolean parameter.
	Whether to generate authoring that relies on
	auto-generation of component GUIDs. The
	default is \$(HarvestAutogenerateGuids) if
	specified; otherwise, true .
GenerateGuidsNow	
	Whether to generate authoring that generates
	durable GUIDs when harvesting. The default is
	\$(HarvestGenerateGuidsNow) if specified;
	otherwise, false .
OutputFile	Required item parameter.
	Specifies the output file that contains the
	generated authoring.
SuppressFragments	Optional boolean parameter.
	parameter.
	Whether to suppress generation of separate
	fragments when harvesting. The default is
	\$(HarvestSuppressFragments) if specified;
	otherwise, true .
SuppressUniquelds	Optional boolean parameter.
	Whether to suppress generation of unique
	component IDs. The default is
	\$(HarvestSuppressUniqueIds) if specified;
	otherwise, false .
Transforms	Optional string parameter.
	XSL transforms to apply to all generated WiX
	authoring. Separate multiple transforms with
	semicolons. The default is \$(HarvestTransforms)
	if specified.

The following table describes the parameters that are specific to the **HeatFile** task.

Parameter	Description
File	Required item group parameter.
	The list of files to harvest.
ComponentGroupName	Optional string parameter.
	The name of the ComponentGroup to create for all the generated authoring.
DirectoryRefld	Optional string parameter.
	The ID of the directory to reference instead of TARGETDIR.
PreprocessorVariable	Optional string parameter.
	Substitute SourceDir for another variable name (ex: var.Dir).
SuppressCom	Optional boolean parameter.
	Suppress generation of COM registry elements. The default is false .
SuppressRegistry	Optional boolean parameter.
	Suppress generation of a Directory element for the parent directory of the file. The default is false .
SuppressRootDirectory	Optional boolean parameter.
	Suppress generation of any registry elements. The default is false .

HeatProject Task

The **HeatProject** task wraps heat.exe, the WiX harvester, using the **project** harvesting type. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the **HeatProject** task:

```
<HeatProject
NoLogo="$(HarvestProjectsNoLogo)"
SuppressAllWarnings="$(HarvestProjectsSuppressAllWarnings)"
SuppressSpecificWarnings="$(HarvestProjectsSuppressSpecificWarnings)"
ToolPath="$(WixToolPath)"
TreatWarningsAsErrors="$(HarvestProjectsTreatWarningsAsErrors)"
TreatSpecificWarningsAsErrors="$(HarvestProjectsTreatSpecificWarningsAsE
VerboseOutput="$(HarvestProjectsVerboseOutput)"
AutogenerateGuids="$(HarvestProjectsAutogenerateGuids)"
GenerateGuidsNow="$(HarvestProjectsGenerateGuidsNow)"
OutputFile="$(IntermediateOutputPath) %( Project.Filename).wxs"
SuppressFragments="$(HarvestProjectsSuppressFragments)"
SuppressUniqueIds="$(HarvestProjectsSuppressUniqueIds)"
Transforms="%(_Project.Transforms)"
Project="@(_Project)"
ProjectOutputGroups="%(_Project.ProjectOutputGroups)" />
```

The following table describes the common WiX MSBuild parameters that are applicable to the **HeatProject** task.

Parameter	Description
NoLogo	Optional boolean parameter.
	Specifies that the tool logo should be suppressed. The default is false . This is equivalent to the -nologo switch.

SuppressAllWarnings	Optional boolean parameter.	
	Specifies that all warnings should be suppressed. The default is false . This is equivalent to the -sw switch.	
SuppressSpecificWarnings	Optional string parameter.	
	Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.	
TreatSpecificWarningsAsErrors	Optional string parameter.	
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.	
TreatWarningsAsErrors	Optional boolean parameter.	
	Specifies that all warnings should be treated as errors. The default is false . This is equivalent to the -wx switch.	
VerboseOutput	Optional boolean parameter.	
	Specifies that the tool should provide verbose output. The default is false . This is equivalent to the -v switch.	

The following table describes the parameters that are common to all heat tasks that are applicable to the **HeatProject** task.

Parameter	Description	
AutogenerateGuids	Optional boolean parameter.	
	Whether to generate authoring that relies on auto-generation of component GUIDs. The default is \$(HarvestAutogenerateGuids) if	

	specified; otherwise, true .	
GenerateGuidsNow	Optional boolean parameter.	
	Whether to generate authoring that generates durable GUIDs when harvesting. The default is \$(HarvestGenerateGuidsNow) if specified; otherwise, false .	
OutputFile	Required item parameter.	
	Specifies the output file that contains the generated authoring.	
SuppressFragments	S Optional boolean parameter.	
	Whether to suppress generation of separate fragments when harvesting. The default is \$(HarvestSuppressFragments) if specified; otherwise, true .	
SuppressUniquelds	Optional boolean parameter.	
	Whether to suppress generation of unique component IDs. The default is \$(HarvestSuppressUniqueIds) if specified; otherwise, false .	
Transforms	Optional string parameter.	
	XSL transforms to apply to all generated WiX authoring. Separate multiple transforms with semicolons. The default is \$(HarvestTransforms) if specified.	

The following table describes the parameters that are specific to the **HeatProject** task.

Parameter	Description	
Project	Required item group parameter.	

	The list of projects to harvest.
ProjectOutputGroups Optional string parameter.	
	The project output groups to harvest. Separate multiple output groups with semicolons. Examples include "Binaries" and "Source".

Light Task

The Light task wraps <u>light.exe</u>, the WiX linker. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the Light task:

The following table describes the common WiX MSBuild parameters that are applicable to the **Light** task.

Parameter	Description
BaseInputPaths	Optional string parameter.
	Specifies a base path that should be used to locate all files. This is equivalent to the -b <path> switch.</path>
BindFiles	Optional boolean parameter.
	Specifies that the tool should bind files into a .wixout file. This is only valid when the OutputAsXml parameter is also provided. This is

	equivalent to the -bf switch.
Pedantic	Optional boolean parameter.
	Specifies that the tool should display pedantic messages. This is equivalent to the -pedantic switch.
SuppressAllWarnings	Optional boolean parameter.
	Specifies that all warnings should be suppressed. This is equivalent to the -sw switch.
SuppressIntermediateFileVersionMatching	Optional boolean parameter.
	Specifies that the tool should suppress intermediate file version mismatch checking. This is equivalent to the -sv switch.
SuppressSchemaValidation	Optional boolean parameter.
	Specifies that schema validation of documents should be suppressed. This is equivalent to the -ss switch.
SuppressSpecificWarnings	Optional string parameter.

	Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.
TreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.
TreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all warnings should be treated as errors. This is equivalent to the -wx switch.
VerboseOutput	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the ${\bf Light}$ task.

Parameter	Description
AllowIdenticalRows	Optional boolean parameter.
	Specifies that the lir

AllowUnresolvedReferences	should allow identic rows. Identical rows be treated as warnin This is equivalent to ai switch in light.exe Optional boolean parameter.
	Specifies that the lir should allow unreso references. This will create valid output. equivalent to the -au switch in light.exe.
AdditionalCub	Optional string parameter.
	Specifies an additio .cub file that the link should use when ru ICE validation. This equivalent to the -cu <file.cub> switch in light.exe.</file.cub>
BackwardsCompatibleGuidGeneration	Optional boolean parameter.
	Specifies that the lir should use the back compatible GUID generation algorithn is equivalent to the switch in light.exe.
CabinetCachePath	Optional string parameter.
	Specifies a path tha linker should use to

	built cabinet files. The equivalent to the -co <path> switch in light</path>
CabinetCreationThreadCount	Optional integer parameter.
	Specifies that numb threads that the link should use when bu cabinet files. This is equivalent to the -ct switch in light.exe.
Cultures	Optional string parameter.
	Specifies a semicolocomma delimited lis localized string cultuload from .wxl files a libraries. Precedence cultures is from left right. This is equival the -cultures: <cultures in="" light.exe.<="" th="" witch=""></cultures>
DefaultCompressionLevel	Optional string parameter.
	Specifies the compression level the linker should use whe building cabinet files Valid values are low medium, high, none mszip. This is equivato the -dcl: <level> strip light.exe.</level>
DropUnrealTables	Optional boolean parameter.

	Specifies that the lir should drop unreal t from the output ima This is equivalent to dut switch in light.ex
ExactAssemblyVersions	Optional boolean parameter.
	Specifies that the lir should use exact assembly versions. is equivalent to the switch in light.exe.
Ices	Optional string parameter.
	Specifies that the lir should run specific internal consistency evaluators (ICEs). T equivalent to the -ic <ice> switch in ligh</ice>
LeaveTemporaryFiles	Optional boolean parameter.
	Specifies that the lir should not delete temporary files. This equivalent to the -no switch in light.exe.
LinkerAdditionalOptions	Optional string parameter.
	Specifies additional command line parar to append when cal light.exe.

LinkerBaseInputPaths	Optional string parameter.
	Specifies a base pa the linker should us locate all files. This equivalent to the -b <path> switch in ligh</path>
LinkerBindFiles	Optional boolean parameter.
	Specifies that the lir should bind files into wixout file. This is covalid when the OutputAsXml parance is also provided. The equivalent to the -bf switch in light.exe.
LinkerPedantic	Optional boolean parameter.
	Specifies that the lir should display peda messages. This is equivalent to the -pedantic switch in light.exe.
LinkerSuppressAllWarnings	Optional boolean parameter.
	Specifies that all linl warnings should be suppressed. This is equivalent to the -su switch in light.exe.
LinkerSuppressIntermediateFileVersionMatching	Optional boolean

	parameter.
	Specifies that the lir should suppress intermediate file ver mismatch checking. is equivalent to the switch in light.exe.
LinkerSuppressSchemaValidation	Optional boolean parameter.
	Specifies that the lir should suppress scl validation of docum This is equivalent to ss switch in light.exe
LinkerSuppressSpecificWarnings	Optional string parameter.
	Specifies that certai linker warnings shous suppressed. This is equivalent to the -suswitch in light.exe.
LinkerTreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certai linker warnings shout treated as errors. The equivalent to the -waswitch in light.exe.
LinkerTreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all linl warnings should be treated as errors. Th

	equivalent to the -w. switch in light.exe.
LinkerVerboseOutput	Optional boolean parameter.
	Specifies that the lir should provide verb output. This is equiv to the -v switch in light.exe.
OutputAsXml	Optional boolean parameter.
	Specifies that the lir should output a .wix file instead of a .msi This is equivalent to xo switch in light.ex
PdbOutputFile	Optional string parameter.
	Specifies that the lir should create the ou wixpdb file with the provided name. This equivalent to the -po <output.wixpdb> sw light.exe.</output.wixpdb>
ReuseCabinetCache	Optional boolean parameter.
	Specifies that the lir should reuse cabine from the cabinet cac This is equivalent to reusecab switch in light.exe.

SetMsiAssemblyNameFileVersion	Optional boolean parameter.
	Specifies that the lir should add a fileVer entry to the MsiAssemblyName for each assembly. equivalent to the -fv switch in light.exe.
SuppressAcIReset	Optional boolean parameter.
	Specifies that the lir should suppress res ACLs. This is useful laying out an image network share. This equivalent to the -sa switch in light.exe.
SuppressAssemblies	Optional boolean parameter.
	Specifies that the lir should not get asse name information fo assemblies. This is equivalent to the -sa switch in light.exe.
SuppressDefaultAdminSequenceActions	Optional boolean parameter.
	Specifies that the lir should suppress de admin sequence ac This is equivalent to sadmin switch in lig
SuppressDefaultAdvSequenceActions	Optional boolean

	parameter.
	Specifies that the lir should suppress de advertised sequenc actions. This is equi to the -sadv switch i light.exe.
SuppressDefaultUISequenceActions	Optional boolean parameter.
	Specifies that the lir should suppress de UI sequence actions is equivalent to the switch in light.exe.
SuppressFileHashAndInfo	Optional boolean parameter.
	Specifies that the lir should suppress gathering file inform (hash, version, lang etc). This is equivalenthe -sh switch in ligh
SuppressFiles	Optional boolean parameter.
	Specifies that the lir should suppress gathering all file dat This has the same as setting the SuppressAssemblie SuppressFileHashA parameters. This is equivalent to the -sf switch in light.exe.

Suppressices	Optional string parameter.
	Specifies that the lir should suppress rur specific ICEs. This i equivalent to the -si <ice> switch in ligh</ice>
SuppressLayout	Optional boolean parameter.
	Specifies that the lir should suppress lay creation. This is equivalent to the -sl switch in light.exe.
SuppressMsiAssemblyTableProcessing	Optional boolean parameter.
	Specifies that the lir should suppress processing the data MsiAssembly table is equivalent to the switch in light.exe.
SuppressPdbOutput	Optional boolean parameter.
	Specifies that the lir should suppress outputting .wixpdb firthis is equivalent to spdb switch in light.
SuppressValidation	Optional boolean parameter.
	Specifies that the lir should suppress .m

	.msm validation. Th equivalent to the -s\ switch in light.exe.
SuppressTagSectionIdAttributeOnTuples	Optional boolean parameter.
	Specifies that the lir should suppress ad the sectionId attribu rows. This is equiva the -sts switch in light.exe.
UnreferencedSymbolsFile	Optional string parameter.
	Specifies an unrefer symbols file that the should use. This is equivalent to the -us <output.xml> switch light.exe.</output.xml>
WixVariables	Optional string parameter.
	Specifies a semicolong delimited list of bind WiX variables. This equivalent to the dename [= < value > switch in light.exe.

Lit Task

The Lit task wraps <u>lit.exe</u>, the WiX library creation tool. It supports a variety of settings that are described in more detail below. To control these settings in your .wixproj file, you can create a PropertyGroup and specify the settings that you want to use for your build process. The following is a sample PropertyGroup that contains settings that will be used by the Lit task:

The following table describes the common WiX MSBuild parameters that are applicable to the **Lit** task.

Parameter	Description
BindFiles	Optional boolean parameter.
	Specifies that the tool should bind files into a .wixout file. This is only valid when the OutputAsXml parameter is also provided. This is equivalent to the -bf switch.
Pedantic	Optional boolean parameter. Specifies that the tool should display pedantic messages. This is equivalent to the -

	pedantic switch.
SuppressAllWarnings	Optional boolean
	parameter.
	Specifies that all
	warnings should be
	suppressed. This is
	equivalent to the -sw
	switch.
SuppressIntermediateFileVersionMatching	
	parameter.
	Specifies that the tool
	should suppress
	intermediate file version
	mismatch checking. This is equivalent to the -sv
	switch.
Cuppus a Cabama Validation	
SuppressSchemaValidation	Optional boolean parameter.
	parameter.
	Specifies that schema
	validation of documents
	should be suppressed.
	This is equivalent to the -
	ss switch.
SuppressSpecificWarnings	Optional string
	parameter.
	Specifies that certain
	warnings should be
	suppressed. This is
	equivalent to the -sw[N]
	switch.
TreatSpecificWarningsAsErrors	Optional string
	parameter.
	ll l

	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx[N] switch.
TreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all warnings should be treated as errors. This is equivalent to the -wx switch.
VerboseOutput	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the $\mbox{\bf Lit}$ task.

Parameter	Description
LibAdditionalOptions	Optional string parameter.
	Specifies additional command line parameters to append when calling lit.exe.
LibBindFiles	Optional boolean parameter.
	Specifies that the library creation tool should bind files into a

LibPedantic	wixout file. This is only valid when the OutputAsXml parameter is also provided. This is equivalent to the -bf switch in lit.exe. Optional boolean parameter.
	Specifies that the library creation tool should display pedantic messages. This is equivalent to the -pedantic switch in lit.exe.
LibSuppressAllWarnings	Optional boolean parameter. Specifies that all library creation tool warnings should be suppressed. This is equivalent to the -sw switch in lit.exe.
LibSuppressIntermediateFileVersionMatching	Optional boolean parameter. Specifies that the library creation tool should suppress intermediate file version mismatch checking. This is equivalent to the -sv switch in lit.exe.
LibSuppressSchemaValidation	Optional boolean

	parameter.
	Specifies that the library creation tool should suppress schema validation of documents. This is equivalent to the -ss switch in lit.exe.
LibSuppressSpecificWarnings	Optional string parameter.
	Specifies that certain library creation tool warnings should be suppressed. This is equivalent to the -sw[N] switch in lit.exe.
LibTreatSpecificWarningsAsErrors	Optional string parameter.
	Specifies that certain library creation tool warnings should be treated as errors. This is equivalent to the -wx[N] switch in lit.exe.
LibTreatWarningsAsErrors	Optional boolean parameter.
	Specifies that all library creation tool warnings should be treated as errors. This is equivalent to the -wx switch in lit.exe.
LibVerboseOutput	Optional boolean

	parameter.
	Specifies that the library creation tool should provide verbose output. This is equivalent to the -v switch in lit.exe.
LinkerBaseInputPaths	Optional string parameter.
	Specifies a base path that the library creation tool should use to locate all files. This is equivalent to the -b <path> switch in lit.exe.</path>

WiX MSBuild Target Reference

This section explains MSBuild targets that are included with the WiX toolset.

- HarvestDirectory Target
- HarvestFile Target
- <u>HarvestProjects Target</u>

HarvestDirectory Target

The **HarvestDirectory** target passes **HarvestDirectory** items to the <u>HeatDirectory task</u> to generate authoring from a file. Authoring is generated for type libraries and self-registration from DllRegisterServer for any files found in directories. This target is processed before compilation. Generated authoring is automatically added to the **Compile** item group to be compiled by the <u>Candle task</u>.

The following tables describe the common WiX MSBuild properties and items that are applicable to the **HarvestDirectory** target.

Items

The following items and item metadata are used by the **HarvestDirectory** target.

Item or Metadata	Description
@(HarvestDirectory)	Required item group.
	The list of directories to harvest.
%	Optional string metadata.
(HarvestDirectory.ComponentGroupName)	
	The name of the
	ComponentGroup to
	create for all the
	generated authoring.
%(HarvestDirectory.DirectoryRefId)	Optional string metadata.
	The ID of the directory to reference instead of TARGETDIR.
% (HarvestDirectory.KeepEmptyDirectories)	Optional boolean metadata.
	Whether to create
	Directory entries for
	empty directories. The
	default is false .
%(HarvestDirectory.PreprocessorVariable)	Optional string metadata.
	Substitute SourceDir for another variable name (ex: var.Dir).
%(HarvestDirectory.SuppressCom)	Optional boolean metadata.

	Suppress generation of COM registry elements. The default is false .
% (HarvestDirectory.SuppressRootDirectory)	Optional boolean metadata.
	Suppress generation of a Directory element for the parent directory of the file. The default is false .
%(HarvestDirectory.SuppressRegistry)	Optional boolean metadata.
	Suppress generation of any registry elements. The default is false .
%(HarvestDirectory.Transforms)	Optional string metadata.
	XSL transforms to apply to the generated WiX authoring. Separate multiple transforms with semicolons.

Properties

The following properties are used by the **HarvestDirectory** target.

Property	Description
\$(HarvestDirectoryAutogenerateGuids)	Optional boolear
	Whether to gener on auto-generatic
	\$(HarvestAutogerotherwise, true .
\$(HarvestDirectoryComponentGroupName)	Optional string p
	The component g contain all genera
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectoryDirectoryRefId)	Optional string p
	The identifier of the that will contain a
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectoryGenerateGuidsNow)	Optional boolear
	Whether to gener generates durable harvesting. The d \$(HarvestGenera specified; otherwi

\$(HarvestDirectoryKeepEmptyDirectories)	Optional boolea n
	Whether to create empty directories default is false .
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectoryNoLogo)	Optional boolea n
	Whether to show The default is \$(N otherwise, false .
\$(HarvestDirectoryPreprocessorVariable)	Optional string p
	Substitute Source name (ex: var.Dir authoring.
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectorySuppressAllWarnings)	Optional boolea n
	Specifies that all suppressed. The \$(HarvestSuppresspecified; otherwi
\$(HarvestDirectorySuppressCom)	Optional boolea n
	Whether to suppr registry elements directories. The d
	MSBuild 4.0 or Vi newer.

\$(HarvestDirectorySuppressFragments)	Optional boolean
	Whether to suppr separate fragmen default is \$(Harve if specified; other
\$(HarvestDirectorySuppressRegistry)	Optional boolean
	Whether to suppr registry elements directories. The d
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectorySuppressRootDirectory)	Optional boolean
	Whether to suppr Directory element harvesting. The d
	This global prope MSBuild 4.0 or Vinewer.
\$(HarvestDirectorySuppressSpecificWarnings)	Optional string p
	Specifies that cer suppressed. The \$(HarvestSuppresspecified.
\$(HarvestDirectorySuppressUniquelds)	Optional boolea n
	Whether to suppr unique componer \$(HarvestSuppre: specified; otherwi
\$(HarvestDirectoryTransforms)	Optional string p

	XSL transforms to WiX authoring. So transforms with so is \$(HarvestTrans
	This global prope MSBuild 4.0 or Vi newer.
\$(HarvestDirectoryTreatSpecificWarningsAsErrors)	Optional string p
	Specifies that cer treated as errors. \$(HarvestTreatSpif specified.
\$(HarvestDirectoryTreatWarningsAsErrors)	Optional boolea n
	Specifies that all treated as errors. \$(HarvestTreatWaspecified; otherwi
\$(HarvestDirectoryVerboseOutput)	Optional boolea n
	Specifies that the verbose output. T \$(HarvestVerbose otherwise, false.

HarvestFile Target

The **HarvestFile** target passes **HarvestFile** items to the <u>HeatFile task</u> to generate authoring from a file. Authoring is generated for type libraries and self-registration from DllRegisterServer. This target is processed before compilation. Generated authoring is automatically added to the **Compile** item group to be compiled by the <u>Candle task</u>.

The following tables describe the common WiX MSBuild properties and items that are applicable to the **HarvestFile** target.

Items

The following items and item metadata are used by the $\mbox{\bf HarvestFile}$ target.

Item or Metadata	Description
@(HarvestFile)	Required item group.
	The list of files to harvest.
%	Optional string metadata.
(HarvestFile.ComponentGroupName)	
	The name of the
	ComponentGroup to create for
	all the generated authoring.
%(HarvestFile.DirectoryRefId)	Optional string metadata.
	The ID of the directory to
	reference instead of
	TARGETDIR.
%(HarvestFile.PreprocessorVariable)	Ontional string metadata
	Optional String metadata.
	Substitute SourceDir for
	another variable name (ex:
	var.Dir).
%(HarvestFile.SuppressCom)	Optional boolean metadata.
	Suppress generation of COM
	registry elements. The default
	is false .
%	Optional boolean metadata.
(HarvestFile.SuppressRootDirectory)	Suppress generation of a
	Directory element for the parent
	directory of the file. The default
	is false .

%(HarvestFile.SuppressRegistry)	Optional boolean metadata. Suppress generation of any registry elements. The default
%(HarvestFile.Transforms)	is false . Optional string metadata.
	XSL transforms to apply to the generated WiX authoring. Separate multiple transforms with semicolons.

Properties

The following properties are used by the **HarvestFile** target.

Property	Description
\$(HarvestFileAutogenerateGuids)	Optional boolean prope
	Whether to generate au
	on auto-generation of c
	The default is
	\$(HarvestAutogenerate
	otherwise, true .
\$(HarvestFileComponentGroupName)	Optional string propert
	The component group r
	contain all generated au
	This global property is (
	MSBuild 4.0 or Visual S
	newer.
\$(HarvestFileDirectoryRefId)	Optional string propert
	The identifier of the Dire
	that will contain all gene
	This global property is (
	MSBuild 4.0 or Visual S
	newer.
\$(HarvestFileGenerateGuidsNow)	Optional boolean prope
	Whether to generate au
	generates durable GUII
	harvesting. The default
	\$(HarvestGenerateGuid
	specified; otherwise, fa
	i i

\$(HarvestFileNoLogo)	Optional boolean prope
	Whether to show the log The default is \$(NoLogo otherwise, false .
\$(HarvestFilePreprocessorVariable)	Optional string propert
	Substitute SourceDir fo name (ex: var.Dir) in all authoring.
	This global property is (MSBuild 4.0 or Visual S newer.
\$(HarvestFileSuppressAllWarnings)	Optional boolean parar
	Specifies that all warnir suppressed. The defaul \$(HarvestSuppressAllW specified; otherwise, fa ll
\$(HarvestFileSuppressCom)	Optional boolean prope
	Whether to suppress ge registry elements when default is false .
	This global property is a MSBuild 4.0 or Visual S newer.
\$(HarvestFileSuppressFragments)	Optional boolean prope
	Whether to suppress ge separate fragments who default is \$(HarvestSup if specified; otherwise, 1
\$(HarvestFileSuppressRegistry)	Optional boolean prope

	Whether to suppress geregistry elements when default is false . This global property is companied to MSBuild 4.0 or Visual S
	newer.
\$(HarvestFileSuppressRootDirectory)	Optional boolean prope
	Whether to suppress ge Directory element for al harvesting. The default This global property is companied to MSBuild 4.0 or Visual S
	newer.
\$(HarvestFileSuppressSpecificWarnings)	Optional string parame
*(·······go,	
	Specifies that certain w
	suppressed. The defaul
	\$(HarvestSuppressSpe specified.
\$(HarvestFileSuppressUniquelds)	Optional boolean prope
φ(narvestriiesappressoniqueius)	Optional boolean prope
	Whether to suppress ge
	unique component IDs.
	\$(HarvestSuppressUnic specified; otherwise, fa
\$(HarvestFileTransforms)	Optional string propert
φ(narvestriie narisionns)	Optional string propert
	XSL transforms to apply
	WiX authoring. Separat
	transforms with semicol
	is \$(HarvestTransforms
	This global property is c MSBuild 4.0 or Visual S newer.

\$(HarvestFileTreatSpecificWarningsAsErrors)	Optional string parame
	Specifies that certain w treated as errors. The d \$(HarvestTreatSpecific\ if specified.
\$(HarvestFileTreatWarningsAsErrors)	Optional boolean parar
	Specifies that all warnir treated as errors. The d \$(HarvestTreatWarning specified; otherwise, fa l
\$(HarvestFileVerboseOutput)	Optional boolean parar
	Specifies that the tool s verbose output. The de \$(HarvestVerboseOutpl otherwise, false .

HarvestProjects Target

The **HarvestProjects** target passes **HarvestProject** items to the <u>HeatProject task</u> to generate authoring from a project file. This target is processed before compilation. Generated authoring is automatically added to the **Compile** item group to be compiled by the <u>Candle task</u>.

```
<ItemGroup>
    <HeatProject Include="...\TestProject\TestProject.csproj" >
        <ProjectOutputGroups>Binaries;Sources</ProjectOutputGroups>
        </HeatProject>
        </ItemGroup>
```

The following tables describe the common WiX MSBuild properties and items that are applicable to the **HarvestProjects** target.

Items

The following items and item metadata are used by the **HarvestProjects** target.

Item or Metadata	Description
@(HarvestProject)	Required item group.
	The list of projects to harvest. The HeatProject item group is provided only for backward compatibility.
% (HarvestProject.ProjectOutputGroups	Optional string metadata.
	The project output groups to harvest. Separate multiple output groups with semicolons. Examples include "Binaries" and "Source".
%(HarvestProject.Transforms)	Optional string metadata. XSL transforms to apply to the generated WiX authoring. Separate multiple transforms with semicolons.

Properties

The following properties are used by the **HarvestProjects** target.

escription
otional boolean
hether to genera a auto-generation ne default is HarvestAutogen herwise, true .
otional boolean
hether to genera enerates durable ervesting. The de HarvestGenerate ecified; otherwis
otional boolean
hether to show the default is \$(No herwise, false .
otional string pro
ne project output projects. Separa Itput groups with
nis global proper SBuild 4.0 or Vis wer.
otional boolean

	Specifies that all w suppressed. The c \$(HarvestSuppres specified; otherwis
\$(HarvestProjectsSuppressFragments)	Optional boolean
	Whether to suppre separate fragment default is \$(Harves if specified; otherw
\$(HarvestProjectsSuppressSpecificWarnings)	Optional string pa
	Specifies that certa suppressed. The c \$(HarvestSuppres specified.
\$(HarvestProjectsSuppressUniquelds)	Optional boolean
	Whether to suppre unique component \$(HarvestSuppres specified; otherwis
\$(HarvestProjectsTransforms)	Optional string pro
	XSL transforms to WiX authoring. Se transforms with se is \$(HarvestTransf
	This global proper MSBuild 4.0 or Vis newer.
\$(HarvestProjectsTreatSpecificWarningsAsErrors)	Optional string pa
	Specifies that certa treated as errors. ⁻ \$(HarvestTreatSpe

	if specified.
\$(HarvestProjectsTreatWarningsAsErrors)	Optional boolean
	Specifies that all w treated as errors. ⁻ \$(HarvestTreatWa specified; otherwis
\$(HarvestProjectsVerboseOutput)	Optional boolean
	Specifies that the t verbose output. Th \$(HarvestVerbose)

Using WiX With NAnt

To include the NAnt build tasks in you NAnt project include the following code:

For more information see the following topics:

- Candle task
- Light task
- Lit Task

WiX NAnt Task Reference

This section explains MSBuild tasks that are included with the WiX toolset.

- Candle Task
- Light Task
- Lit Task

Candle Task

The Candle task wraps <u>candle.exe</u>, the WiX compiler. It supports a variety of settings that are described in more detail below. The following is a sample shows a the NAnt code used to run the Candle task:

```
<candle
 exedir="${wix.dir}"
 out="${release.dir}\obj\\"
 rebuild="true"
 extensions="WixUIExtension; WixUtilExtension"
 warningsaserrors="true">
<defines>
  <define name="ProjectDir" value="${release.dir}" />
 <define name="Configuration" value="Release" />
  <define name="Version" value="${buildnumber.version}" />
</defines>
<sources basedir="${releasemsi.dir}">
  <include name="Product.wxs"/>
  <include name="Files.wxs" />
</sources>
</candle>
```

The following table describes the common WiX NAnt parameters that are applicable to the **Candle** task.

Parameter	Description
exedir	Optional string parameter.
	Sets the directory to the tool executable. Defaults to the path specified by the registry key "HKLM\SOFTWARE\Microsoft\Windows Installer XML\3.0\InstallRoot" which is set by the WiX installation. If no path is found or specified the task assumes the executable is on the path.
out	Required string parameter.
	Sets the file or directory to write the output to. This is

	equivalent to the -out switch.
extensions	Optional string parameter.
	Semi-colon separated list of WiX extensions to load. This is equivalent to the -ext switch.
rebuild	Optional boolean parameter.
	Instructs NAnt to recompile the output file regardless of the file timestamps
sources	Required NAnt fileset .
	The set of source files for compilation.
warningsaserrors	Optional boolean parameter.
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx switch.
verbose	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the **Candle** task.

Parameter	Description
defines	Optional NAnt fileset .
	This is equivalent to the -d <name>[=<value>] switch in candle.exe.</value></name>
includedirs	Optional NAnt fileset .
	Specifies directories to add to the compiler include search path. This is equivalent to the -l <dir> switch in candle.exe.</dir>

Additional options can be added by using the standard NAnt **<arg>** elements, e.g.:

<arg line="-pedantic" />

Light Task

The Light task wraps <u>light.exe</u>, the WiX linker. It supports a variety of settings that are described in more detail below. The following is a sample shows a the NAnt code used to run the Light task:

```
exedir="${wix.dir}"
  out="${output.dir}\Setup.msi"
  warningsaserrors="true"
  suppressices="ICE57"
  cultures="en-us"
  extensions="WixUIExtension"
  rebuild="true"
  suppresspdb="true">
  <!-- Specify additional options -->
  <arg line="-fv" />
  <sources basedir="${release.dir}\Setup\obj">
    <include name="**.wixobj" />
  </sources>
```

The following table describes the common WiX NAnt parameters that are applicable to the **Light** task.

Parameter	Description
exedir	Optional string parameter.
	Sets the directory to the tool executable. Defaults to the path specified by the registry key "HKLM\SOFTWARE\Microsoft\Windows Installer XML\3.0\InstallRoot" which is set by the WiX installation. If no path is found or specified the task assumes the executable is on the path.
out	Required string parameter.
	Sets the file or directory to write the output to. This is

	equivalent to the -out switch.
extensions	Optional string parameter.
	Semi-colon separated list of WiX extensions to load. This is equivalent to the -ext switch.
rebuild	Optional boolean parameter.
	Instructs NAnt to recompile the output file regardless of the file timestamps
sources	Required NAnt fileset .
	The set of source files for compilation. This is equivalent to the -xx switch.
warningsaserrors Optional boolean parameter.	
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx switch.
verbose	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the **Light** task.

Parameter	Description
cultures	Optional string parameter.
	Specifies a semicolon-delimited list of localized string cultures that the linker should load from libraries. This is equivalent to the -cultures switch in light.exe.
localizations	Optional NAnt fileset .
	The set of localization files (.wxl) to include. This is equivalent to the -loc switch in light.exe.
suppressices	Optional string parameter.

	Specifies a semicolon-delimited list of ICE validations that the linker should suppress running. This is equivalent to the -sice: <ice> switch in light.exe.</ice>
suppresspdb	Optional boolean parameter.
	Specifies that the linker should suppress outputting .wixpdb files. This is equivalent to the -spdb switch in light.exe.
reusecab	Optional boolean parameter.
	Specifies that the linker should reuse cabinet files from the cabinet cache. This is equivalent to the -reusecab switch in light.exe.
cabcache	Optional string parameter.
	Specifies a path that the linker should use to cache built cabinet files. This is equivalent to the -cc switch in light.exe.
fileversions	Optional boolean parameter.
	Specifies that the linker should add a 'fileVersion' entry to the MsiAssemblyName table. This is equivalent to the -fv switch in light.exe.

Additional options can be added by using the standard NAnt <arg>elements, e.g.:

```
<arg line="-sacl" />
```

Lit Task

The Light task wraps <u>lit.exe</u>, the WiX library tool. It supports a variety of settings that are described in more detail below. The following is a sample shows a the NAnt code used to run the Lit task:

The following table describes the common WiX NAnt parameters that are applicable to the **Lit** task.

Parameter	Description
exedir	Optional string parameter.
	Sets the directory to the tool executable. Defaults to the path specified by the registry key "HKLM\SOFTWARE\Microsoft\Windows Installer XML\3.0\InstallRoot" which is set by the WiX installation. If no path is found or specified the task assumes the executable is on the path.
out	Required string parameter.
	Sets the file or directory to write the output to. This is equivalent to the -out switch.
extensions	Optional string parameter.
	Semi-colon separated list of WiX extensions to load. This is equivalent to the -ext switch.
4	ll l

rebuild	Optional boolean parameter.
	Instructs NAnt to recompile the output file regardless of the file timestamps
sources	Required NAnt fileset .
	The set of source files for compilation. This is equivalent to the -xx switch.
warningsaserrors	Optional boolean parameter.
	Specifies that certain warnings should be treated as errors. This is equivalent to the -wx switch.
verbose	Optional boolean parameter.
	Specifies that the tool should provide verbose output. This is equivalent to the -v switch.

The following table describes the parameters that are specific to the **Lit** task.

Parameter	Description
bindfiles	Optional boolean parameter.
	This is equivalent to the -bf switch in lit.exe.
localizations	Optional NAnt fileset .
	The set of localization files (.wxl) to include. This is equivalent to the -loc switch in lit.exe.

Additional options can be added by using the standard NAnt **<arg>** elements, e.g.:

<arg line="-ss" />

How To Guides

This section includes How To documentation for performing common WiX tasks.

Files, Shortcuts and Registry

- Add a file to your installer
- Check the version number of a file during installation
- Write a registry entry during installation
- Read a registry entry during installation
- Create a shortcut on the Start Menu
- Create a shortcut to a web page
- Create an uninstall shortcut
- NGen managed assemblies during installation
- Reference another DirectorySearch element
- Get the parent directory of a file search

Redistributables and Install Checks

- Check for .NET Framework versions
- Install the .NET Framework using a bootstrapper
- Install DirectX 9.0 with your installer
- Install the Visual C++ Redistributable with your installer
- Block installation based on OS version

User Interface and Localization

- Build a localized version of your installer
- Make your installer localizable
- Run the installed application after setup
- Set your installer's icon in Add/Remove Programs

Product Updates

• Implement a major upgrade in your installer

Others

- Get a log of your installation for debugging
- Look inside your MSI with Orca
- Generate a GUID
- <u>Use WiX Extensions</u>
- Optimize building cabinet files
- Specify source file locations

How To: Files, Shortcuts and Registry

This section includes how to guides that demonstrate how to work with files, shortcuts, and the Windows registry.

- How To: Add a file to your installer
- How To: Check the version number of a file during installation
- How To: Write a registry entry during installation
- How To: Read a registry entry during installation
- How To: Create a shortcut on the Start Menu
- How To: Create a shortcut to a web page
- How To: Create an uninstall shortcut
- How To: NGen managed assemblies during installation
- How To: Reference another DirectorySearch element

How To: Add a File To Your Installer

Installing files is the most fundamental aspect of any installer, and is usually what leads people to build an installer in the first place. Learning how to place a file on disk using Windows Installer best practices not only ensures maintainability going forward, but also enables you to build patches later if necessary.

Step 1: Define the directory structure

Installers frequently have many files to install into a few locations on disk. To improve the readability of the WiX file, it is a good practice to define your installation directories first before listing the files you'll install. Directories are defined using the Directory element and describe the hierarchy of folders you would like to see on the target machine. The following sample defines a directory for the installation of the main application executable.

The element with the id <u>TARGETDIR</u> is required by the Windows Installer and is the root of all directory structures for your installation. Every WiX project will have this directory element. The second element, with the id <u>ProgramFilesFolder</u>, uses a pre-defined Windows Installer property to reference the Program Files folder on the user's machine. In most cases this will resolve to **c:\Program Files**. The third directory element creates your application's folder under Program Files, and it is given the id APPLICATIONROOTDIRECTORY for later use in the WiX project. The id is in all capital letters to make it a <u>public property</u> that can be set from UI or via the command line.

The result of these tags is a **c:\Program Files\My Application Name** folder on the target machine.

Step 2: Add files to your installer package

A file is added to the installer using two elements: a Component element to specify an atomic unit of installation and a File element to specify the file that should be installed.

The component element describes a set of resources (usually files, registry entries, and shortcuts) that need to be installed as a single unit. This is separate from whether the set of items consist of a logical feature the user can select to install which is discussed in Step 3. While it may not seem like a big deal when you are first authoring your installer, components play a critical role when you decide to build patches at a later date.

In general, you should restrict yourself to a single file per component. The Windows Installer is designed to support thousands of components in a single installer, so unless you have a very good reason, keep to one file per component. Every component must have its own unique GUID. Failure to follow these two basic rules can lead to many problems down the road when it comes to servicing.

The following sample uses the directory structure defined in Step 1 to install two files: an application executable and a documentation file.

The <<u>DirectoryRef</u>> element is used to refer to the directory structure created in step 1. By referencing the APPLICATIONROOTDIRECTORY directory, the files will be installed into the **c:\program files\My Application Name** folder. Underneath the DirectoryRef are two Component elements, one for each of the two files that will be installed. This is in keeping with the best practice of having one component per file.

Each Component element is given an Id and a Guid. The Id is used to refer to the component later in the WiX project. The Guid is used later for patches and must be unique for each component. For information on generating GUIDs see How To: Generate a GUID.

Beneath each component is a File element that does the actual work of packaging your source files into the installer. The Id is used to refer to the file elsewhere in the WiX project. The Source attribute specifies the location of the file on your machine, so WiX can find it and build it into the installer.

The KeyPath attribute is set to yes to tell the Windows Installer that this particular file should be used to determine whether the component is installed. When you have one file per component you should always set the KeyPath attribute to yes. The Checksum attribute should be set to yes for executable files that have a checksum value in the file header (this is generally true for all executables), and is used by the Windows Installer to verify the validity of the file on re-install.

Step 3: Tell Windows Installer to install the files

After defining the directory structure and listing the files to package into the installer, the last step is to tell Windows Installer to actually install the files. The Feature element is used to do this, and is where you break up your installer into logical pieces that the user can install independently. The following example creates a single feature that installs the application executable and documentation from Step 2.

```
<Feature Id="MainApplication" Title="Main Application" Level="1">
        <ComponentRef Id="myapplication.exe" />
        <ComponentRef Id="documentation.html" />
        </Feature>
```

The Feature is given a Id. If you are using an installer UI sequence that includes feature selection, the Title attribute contains the text displayed in the UI for the feature. The Level attribute should be set to 1 to enable the installation of the feature by default.

The <<u>ComponentRef</u>> element is used to reference the components created in Step 2 via the Id attribute.

The Complete Sample

The following is a complete sample that uses the above concepts. This example can be inserted into a WiX project and compiled, or compiled and linked from the command line, to generate an installer.

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" UpgradeCode="PUT-GUID-HERE" Version="1.0.0.0" Language="1033" Name="My A
    <Package InstallerVersion="300" Compressed="yes"/>
    <Media Id="1" Cabinet="myapplication.cab" EmbedCab="yes" />
    <!-- Step 1: Define the directory structure -->
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Application Name"/>
      </Directory>
    </Directory>
    <!-- Step 2: Add files to your installer package -->
    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\MyApplication.exe" KeyPath="yes" Cher
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE">
        <File Id="documentation.html" Source="MySourceFiles\documentation.html" KeyPath="yes"/>
      </Component>
    </DirectoryRef>
    <!-- Step 3: Tell WiX to install the files -->
    <Feature Id="MainApplication" Title="Main Application" Level="1">
      <ComponentRef Id="myapplication.exe" />
      <ComponentRef Id="documentation.html" />
    </Feature>
  </Product>
</Wix>
```

How To: Check the Version Number of a File During Installation

Installers often need to look up the version number of a file on disk during the installation process. The check is often used in advance of a conditional statement later in install, such as to block the user from installing if a file is missing, or to display custom installation UI depending on whether the file version is high enough. This how to demonstrates verifying the version of a file on disk, then using the resulting property to block the application's installation if the file version is lower than expected.

Step 1: Determine the version of the file

File versions are determined using the Property, DirectorySearch and FileSearch elements. The following snippet looks for the user32.dll file in the machine's System32 directory and checks to see if it is at least version 6.0.6001.1751.

Searching for a file is accomplished by describing the directories to search, and then specifying the file to look up in that directory.

The Property element defines the Id for the results of the file search. This Id is used later in the WiX project, for example in conditions. The DirectorySearch element is used to build the directory hierarchy to search for the file. In this case it is given a unique Id, and the path is set to the Windows Installer defined SystemFolder property which points to the user's Windows\System32 directory. The FileSearch element specifies the name of the file to look for in the parent DirectorySearch folder. The MinVersion attribute specifies the minimum version of the file to find.

If the file is found successfully the USER32VERSION property will be set to the full path to the user32.dll file.

Important: When doing a locale-neutral search for a file, you must set the MinVersion property to one revision number lower than the actual version you want to search for. In this example, while we want to find file version 6.0.6001.1751, the MinVersion is set to 6.0.6001.1750. This is because of a quirk in how the Windows Installer matches file versions. More information is available in the Windows Installer documentation.

Step 2: Use the property in a condition

Once you have determined whether the file exists with the requested version you can use the property in a condition. The following is a simple example that prevents installation of the application if the user32.dll file version is too low.

<Condition Message="The installed version of user32.dll is not high enough to support this installer.">
 <![CDATA[Installed OR USER32VERSION]]>
 </Condition>

Installed is a Windows Installer property that ensures the check is only done when the user is installing the application, rather than on a repair or remove. The USER32VERSION part will pass if the property is set to anything, and will fail if it is not set. The file check in Step 1 will set the property to the full path of the user32.dll file if it is found with an appropriate file version, and will not set it otherwise.

How To: Write a Registry Entry During Installation

Writing registry entries during installation is similar to writing files during installation. You describe the registry hierarchy you want to write into, specify the registry values to create, then add the component to your feature list.

Step 1: Describe the registry layout and values

The following example illustrates how to write two registry entries, one to a specific value and the other to the default value.

The snippet begins with a DirectoryRef that points to the <u>TARGETDIR</u> directory defined by Windows Installer. This effectively means the registry entries should be installed to the target user's machine. Under the DirectoryRef is a Component element that groups together the registry entries to be installed. The component is given an id for reference later in the WiX project and a unique guid.

The registry entries are created by first using the <a href="Regis

HKEY_CURRENT_USER\Software\Microsoft\MyApplicationName. The optional Action attribute is used to tell Windows Installer that the key

should be created (if necessary) on install, and that the key and all its sub-values should be removed on uninstall.

Under the RegistryKey element the RegistryValue element is used to create the actual registry values. The first is the SomeIntegerValue value, which is of type integer and has a value of 1. It is also marked as the KeyPath for the component, which is used by the Windows Installer to determine whether this component is installed on the machine. The second RegistryValue element sets the default value for the key to a string value of Default Value.

The id attribute is omitted on the RegistryKey and RegistryValue elements because there is no need to refer to these items elsewhere in the WiX project file. WiX will auto-generate ids for the elements based on the registry key, value, and parent component name.

Step 2: Tell Windows Installer to install the entries

After defining the directory structure and listing the registry entries to package into the installer, the last step is to tell Windows Installer to actually install the registry entry. The <Feature> element is used to do this. The following snippet adds a reference to the registry entries component, and should be inserted inside a parent Feature element.

<ComponentRef Id="RegistryEntries" />

The ComponentRef element is used to reference the component created in Step 1 via the Id attribute.

How To: Read a Registry Entry During Installation

Installers often need to look up the value of a registry entry during the installation process. The resulting registry value is often used in a conditional statement later in install, such as to install a specific component if a registry entry is not found. This how to demonstrates reading an integer value from the registry and verifying that it exists in a launch condition.

Step 1: Read the registry entry into a property

Registry entries are read using the RegistrySearch> element. The following snippet looks for the presence of the key that identifies the installation of .NET Framework 2.0 on the target machine*.

The RegistrySearch element specifies a unique id, the root in the registry to search, and the key to look under. The name attribute specifies the specific value to query. The type attribute specifies how the value should be treated. Raw indicates that the value should be prefixed according to the data type of the value. In this case, since Install is a DWORD, the resulting value will be prepended with a #.

The above sample will set the NETFRAMEWORK20 property to "#1" if the registry key was found, and to nothing if it wasn't.

Step 2: Use the property in a condition

After the property is set you can use it in a condition anywhere in your WiX project. The following snippet demonstrates how to use it to block installation if .NET Framework 2.0 is not installed.

```
<Condition Message="This application requires .NET Framework 2.0. Please install the .NET Framework 1
<![CDATA[Installed OR NETFRAMEWORK20]]>
</Condition>
```

Installed is a Windows Installer property that ensures the check is only done when the user is installing the application, rather than on a repair or remove. The NETFRAMEWORK20 part of the condition will pass if the property was set. If it is not set the installer will display the error message then abort the installation process.

* This registry entry is used for sample purposes only. If you want to detect the installed version of .NET Framework you can use the built-in WiX support. For more information see How To: Check for .NET Framework Versions.

How To: Create a Shortcut on the Start Menu

When installing applications it is a common requirement to place a shortcut on the user's Start Menu to provide a launching point for the program. This how to walks through how to create a shortcut on the start menu. It assumes you have a WiX source file based on the concepts described in How To: Add a file to your installer.

Step 1: Define the directory structure

Start Menu shortcuts are installed in a different directory than regular application files, so modifications to the installer's directory structure are required. The following WiX fragment should be placed inside a Directory element with the TARGETDIR ID and adds directory structure information for the Start Menu:

```
<Directory Id="ProgramMenuFolder">
        <Directory Id="ApplicationProgramsFolder" Name="My Application Name"/>
</Directory>
```

The <u>ProgramMenuFolder</u> Id is a standard Windows Installer property that points to the Start Menu folder on the target machine. The second Directory element creates a subfolder on the Start Menu called My Application Name, and gives it an id for use later in the WiX project.

Step 2: Add the shortcut to your installer package

A shortcut is added to the installer using three elements: a Component element to specify an atomic unit of installation, a Shortcut element to specify the shortcut that should be installed, and a RemoveFolder element to ensure proper cleanup when your application is uninstalled.

The following sample uses the directory structure defined in Step 1 to create the Start Menu shortcut.

The <<u>DirectoryRef</u>> element is used to refer to the directory structure created in step 1. By referencing the ApplicationProgramsFolder directory the shortcut will be installed into the user's Start Menu inside the My Application Name folder.

Underneath the DirectoryRef is a single Component to group the elements used to install the Shortcut. The first element is Shortcut and it creates the actual shortcut in the Start Menu. The Id attribute is a unique id for the shortcut. The Name attribute is the text that will be displayed in the Start Menu. The description is an optional attribute for an additional application description. The Target attribute points to the executable to launch on disk. Notice how it uses the APPLICATIONROOTDIRECTORY property previously defined in the directory structure. The WorkingDirectory attribute sets the working directory for the shortcut.

To set an optional icon for the shortcut you need to first include the icon

in your installer using the Icon attribute on the Shortcut element.

In addition to creating the shortcut the component contains two other important pieces. The first is a RemoveFolder element, which ensures the ApplicationProgramsFolder is correctly removed from the Start Menu when the user uninstalls the application. The second creates a registry entry on install that indicates the application is installed. This is required as a Shortcut cannot serve as the KeyPath for a component when installing non-advertised shortcuts for the current users. For more information on creating registry entries see How To: Write a registry entry during installation.

Step 3: Tell Windows Installer to install the shortcut

After defining the directory structure and listing the shortcuts to package into the installer, the last step is to tell Windows Installer to actually install the shortcut. The Feature element is used to do this. The following snippet adds a reference to the shortcut component, and should be inserted inside a parent Feature element.

<ComponentRef Id="ApplicationShortcut" />

The ComponentRef element is used to reference the component created in Step 2 via the Id attribute.

The Complete Sample

The following is a complete sample that uses the above concepts. This example can be inserted into a WiX project and compiled, or compiled and linked from the command line, to generate an installer.

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" UpgradeCode="PUT-GUID-HERE" Version="1.0.0.0" Language="1033" Name="My A
    <Package InstallerVersion="300" Compressed="yes"/>
    <Media Id="1" Cabinet="myapplication.cab" EmbedCab="yes" />
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Application Name"/>
      </Directory>
      <!-- Step 1: Define the directory structure -->
      <Directory Id="ProgramMenuFolder">
        <Directory Id="ApplicationProgramsFolder" Name="My Application Name"/>
      </Directory>
    </Directory>
    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\MyApplication.exe" KeyPath="yes" Cher
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE">
        <File Id="documentation.html" Source="MySourceFiles\documentation.html" KeyPath="yes"/>
      </Component>
    </DirectoryRef>
    <!-- Step 2: Add the shortcut to your installer package -->
    <DirectoryRef Id="ApplicationProgramsFolder">
      <Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE">
        <Shortcut Id="ApplicationStartMenuShortcut"</p>
                                                              Name="My Application Name"
          Description="My Application Description"
          Target="[APPLICATIONROOTDIRECTORY]MyApplication.exe"
              WorkingDirectory="APPLICATIONROOTDIRECTORY"/>
        <RemoveFolder Id="ApplicationProgramsFolder" On="uninstall"/>
        < Registry Value Root="HKCU" Key="Software\Microsoft\MyApplicationName" Name="installe
      </Component>
    </DirectoryRef>
    <Feature Id="MainApplication" Title="Main Application" Level="1">
      <ComponentRef Id="myapplication.exe" />
      <ComponentRef Id="documentation.html" />
      <!-- Step 3: Tell WiX to install the shortcut -->
```

```
<ComponentRef Id="ApplicationShortcut" />
</Feature>
</Product>
</Wix>
```

How To: Create a Shortcut to a Webpage

WiX provides support for creating shortcuts to Internet sites as part of the install process. This how to demonstrates referencing the necessary utility library and adding an Internet shortcut to your installer. It assumes you have already followed the steps in the How To: Create a shortcut on the Start Menu.

Step 1: Add the WiX Utility extensions library to your project

The WiX support for Internet shortcuts is included in a WiX extension library that must be added to your project prior to use. If you are using WiX on the command-line you need to add the following to your candle and light command lines:

-ext WiXUtilExtension

If you are using WiX in Visual Studio you can add the extensions using the Add Reference dialog:

- 1. Open your WiX project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the WixUtilExtension.dll assembly from the list and click Add
- 4. Close the Add Reference dialog

Step 2: Add the WiX Utility extensions namespace to your project

Once the library is added to your project, you need to add the Utility extensions namespace to your project so you can access the appropriate WiX elements. To do this modify the top-level <a href="https://www.wix.element.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nlm.ni.nl

xmlns:util="http://schemas.microsoft.com/wix/UtilExtension"

A complete Wix element with the standard namespace and the Utility extensions namespace added looks like this:

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi" xmlns:util="http://schemas.microsoft.com/wix/UtilExtension">

Step 3: Add the Internet shortcut to your installer package

Internet shortcuts are created using the <<u>Util:InternetShortcut></u> element. The following example adds an InternetShortcut element to the existing shortcut creation example from How To: Create a shortcut on the Start Menu.

The InternetShortcut is given a unique id with the Id attribute. in this case the application's Start Menu folder. The Name attribute specifies the name of the shortcut on the Start Menu. The Target attribute specifies the destination address for the shortcut. The DirectoryRef element is used to refer to the directory structure already defined by the project file. By referencing the ApplicationProgramsFolder directory the shortcut will be installed into the user's Start Menu inside the My Application Name folder.

How To: Create an Uninstall Shortcut

When installing an application it is a common requirement to place a shortcut on the user's Start Menu to provide a method of uninstalling the application. This how to demonstrates the steps required to create an uninstall shortcut on the start menu that passes all ICE validation checks.

This how to assumes you are starting with the sample described the <u>How To: Create a Shortcut on the Start Menu</u> topic.

Step 1: Add the Uninstall Shortcut

The <Shortcut> element is used to add the uninstall shortcut to the start menu, and the shortcut points to msiexec.exe (the Windows Installer executable used to actually invoke the uninstall process). Anywhere within the existing ApplicationShortcut component add the following:

```
<Shortcut Id="UninstallProduct"
    Name="Uninstall My Application"
    Target="[System64Folder]msiexec.exe"
    Arguments="/x [ProductCode]"
    Description="Uninstalls My Application" />
```

The Target attribute points to the location of msiexec.exe. The Windows Installer <u>System64Folder</u> property will resolve to the *System32* directory on 32-bit machines and the *SysWow64* directory on 64-bit machines. Using this property ensures msiexec.exe can always be located regardless of the operating system version on the target machine. The Arguments attribute is used to let msiexec.exe know which product to uninstall by passing in the ProductCode for the install package.

To avoid ICE validation errors at build it is important to couple the Shortcut element with a registry entry and a RemoteFolder element. Both of these are described in more detail in the How To: Create a Shortcut on the Start Menu topic, and are shown in the complete sample below.

The Complete Sample

The following is a complete sample that uses the above concepts. This example can be inserted into a WiX project and compiled, or compiled and linked from the command line, to generate an installer.

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="*" UpgradeCode="PUT-GUID-HERE" Version="1.0.0.0" Language="1033" Name="My A
    <Package InstallerVersion="300" Compressed="yes"/>
    <Media Id="1" Cabinet="myapplication.cab" EmbedCab="yes" />
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Application Name"/>
      </Directory>
      <Directory Id="ProgramMenuFolder">
        <Directory Id="ApplicationProgramsFolder" Name="My Application Name"/>
      </Directory>
    </Directory>
    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\MyApplication.exe" KeyPath="yes" Chec</p>
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE">
        <File Id="documentation.html" Source="MySourceFiles\documentation.html" KeyPath="yes"/>
      </Component>
    </DirectoryRef>
    <DirectoryRef Id="ApplicationProgramsFolder">
      Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE">
        <Shortcut Id="ApplicationStartMenuShortcut"</p>
                                                              Name="My Application Name"
          Description="My Application Description"
          Target="[APPLICATIONROOTDIRECTORY]MyApplication.exe"
              WorkingDirectory="APPLICATIONROOTDIRECTORY"/>
        <!-- Step 1: Add the uninstall shortcut to your installer package -->
        <Shortcut Id="UninstallProduct"</p>
              Name="Uninstall My Application"
              Description="Uninstalls My Application"
              Target="[System64Folder]msiexec.exe"
              Arguments="/x [ProductCode]"/>
        <RemoveFolder Id="ApplicationProgramsFolder" On="uninstall"/>
        < Registry Value Root="HKCU" Key="Software\Microsoft\MyApplicationName" Name="installe"
     </Component>
    </DirectoryRef>
```

How To: NGen Managed Assemblies During Installation

NGen during installation can improve your managed application's startup time by creating native images of the managed assemblies on the target machine. This how to describes using the WiX support to NGen managed assemblies at install time.

Step 1: Add the WiX .NET extensions library to your project

The WiX support for NGen is included in a WiX extension library that must be added to your project prior to use. If you are using WiX on the command-line you need to add the following to your candle and light command lines:

-ext WixNetFxExtension

If you are using WiX in Visual Studio you can add the extensions using the Add Reference dialog:

- 1. Open your WiX project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the **WixNetFxExtension.dll** assembly from the list and click Add
- 4. Close the Add Reference dialog

Step 2: Add the WiX .NET extensions namespace to your project

Once the library is added to your project you need to add the .NET extensions namespace to your project so you can access the appropriate WiX elements. To do this modify the top-level <Wix> element in your project by adding the following attribute:

xmlns:netfx="http://schemas.microsoft.com/wix/NetFxExtension"

A complete Wix element with the standard namespace and the .NET extensions namespace added looks like this:

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi" xmlns:netfx="http://schemas.microsoft.com/wix/NetFxExtension">

Step 3: Mark the managed files for NGen

Once you have the .NET extension library and namespace added to your project you can use the NetFx:NativeImage element to enable NGen on your managed assemblies. The NativeImage element goes inside a parent File element:

The Id attribute is a unique identifier for the native image. The Platform attribute specifies the platforms for which the native image should be generated, in this case 32-bit. The Priority attribute specifies when the image generation should occur, in this case immediately during the setup process. The AppBaseDirectory attribute identifies the directory to use to search for dependent assemblies during the image generation. In this case it is set to the install directory for the application.

How To: Reference another DirectorySearch element

There may be times when you need to locate different files or subdirectories under the same directory, and assign each to a separate property. Since you cannot define the same DirectorySearch element more than once, you must use a DirectorySearchRef element. To reference another DirectorySearch element, you must specify the same Id, Parent Id, and Path attribute values or you will get unresolved symbol errors when linking with light.exe.

Step 1: Define a DirectorySearch element

You first need to define the parent DirectorySearch element. This is expected to contain the different files or subdirectories you will assign to separate properties.

This will search for the file "chimes.wav" under the Media directory in Windows. If the file is found, the full path will be assigned to the public property "SHDOCVW".

Step 2: Define a DirectorySearchRef element

To search for another file in the Media directory, you need to reference all the same Id, Parent Id, and Path attributes. Because the Media DirectorySearch element is nested under the WinDir DirectorySearch element, its Parent attribute is automatically assigned the parent DirectorySearch element's Id attribute value; thus, that is what you must specify for the DirectorySearchRef element's Parent attribute value.

If you wanted to refer to another DirectorySearch element that used the Id Media but was under a different parent path, you would have to define a new DirectorySearch element under a different parent than in step 1.

How To: Get the parent directory of a file search

You can set a property to the parent directory of a file.

Step 1: Define the search root

In the following example, the path to [WindowsFolder]Microsoft.NET is defined as the root of the search. If you do not define a search root, Windows Installer will search all fixed drives up to the depth specified.

Step 2: Define the parent directory to find

Under the search root, define the directory you want returned and set the DirectorySearch/@AssignToProperty attribute to 'yes'. You must then define the file you want to find using a unique FileSearch/@Id attribute value.

In this example, if ngen.exe is newer than version 2.0.0.0 and is found no more than two directories under [WindowsFolder]Microsoft.NET its parent directory is returned in the NGEN2DIR property.

How To: Redistributables and Install Checks

This section includes guides for common redistributable installations and pre-installation checks.

- How To: Check for .NET Framework versions
- How To: Install DirectX 9.0 with your installer
- How To: Install the .NET Framework using a bootstrapper
- How To: Install the Visual C++ Redistributable with your installer
- How To: Block installation based on OS version

How To: Check for .NET Framework Versions

When installing applications written using managed code it is often useful to verify that the user's machine has the necessary version of the .NET Framework prior to installation. This how to describes using the WiX support to verify .NET Framework versions at install time. For information on how to install the .NET Framework during your installation see How To: Install the .NET Framework using a bootstrapper.

Step 1: Add the WiX .NET extensions library to your project

The WiX support for NGen is included in a WiX extension library that must be added to your project prior to use. If you are using WiX on the command-line you need to add the following to your candle and light command lines:

-ext WiXNetFxExtension

If you are using WiX in Visual Studio you can add the extensions using the Add Reference dialog:

- 1. Open your WiX project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the **WixNetFxExtension.dll** assembly from the list and click Add
- 4. Close the Add Reference dialog

Step 2: Add the WiX .NET extensions namespace to your project

Once the library is added to your project you need to add the .NET extensions namespace to your project so you can access the appropriate WiX elements. To do this modify the top-level <Wix> element in your project by adding the following attribute:

xmlns:netfx="http://schemas.microsoft.com/wix/NetFxExtension"

A complete Wix element with the standard namespace and the .NET extensions namespace added looks like this:

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi" xmlns:netfx="http://schemas.microsoft.com/wix/NetFxExtension">

Step 3: Reference the required properties in your project

The .NET Framework extensions for WiX define properties for all current versions of the .NET Framework, including service pack levels. To make these properties available to your installer you need to reference them using the <a href="mailto:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spranger:spran

<PropertyRef Id="NETFRAMEWORK20"/>

Step 4: Use the pre-defined properties in a condition

Once the property is referenced you can use it in any WiX condition statement. For example, the following condition blocks installation if .NET Framework 2.0 is not installed.

```
<Condition Message="This application requires .NET Framework 2.0. Please install the .NET Framework t
    <![CDATA[Installed OR NETFRAMEWORK20]]>
```

Installed is a Windows Installer property that ensures the check is only done when the user is installing the application, rather than on a repair or remove. The NETFRAMEWORK20 part of the condition will pass if .NET Framework 2.0 installed. If it is not set the installer will display the error message then abort the installation process.

To check against the service pack level of the framework use the *_SP_LEVEL properties. The following condition blocks installation if .NET Framework 3.0 SP1 is not present on the machine.

```
<Condition Message="This application requires .NET Framework 3.0 SP1. Please install the .NET Framework 3.0 SP1. Please 3.0 SP
```

As with the previous example Installed prevents the check from running when the user is doing a repair or remove. The NETFRAMEWORK30_SP_LEVEL property is set to "#1" if Service Pack 1 is present. Since there is no way to do a numerical comparison against a value with a # in front of it, the condition first checks to see if the NETFRAMEWORK30_SP_LEVEL is set and the confirms that it is set to a number. This will correctly indicate whether any service pack for .NET 3.0 is installed.

How To: Install the .NET Framework Using a Bootstrapper

Applications written using the .NET Framework often need to install the framework as part of their installation process. Due to dependencies in the .NET Framework installer there is no way to include the framework directly within your package (as you can with the <u>Visual C++</u> and <u>DirectX</u> runtimes). Instead you have to rely on a bootstrapper: a wrapper application that first installs the .NET Framework and then runs your application's installer.

WiX does not currently provide a bootstrapper, however you can use the one provided by the ClickOnce deployment features in Visual Studio. This document walks through how to modify a WiX project to generate a ClickOnce bootstrapper for .NET Framework 3.5. Similar steps can be used to generate a bootstrapper for other technologies, such as SQL Server Compact Edition and Visual Studio Tools For Office.

Step 1: Open your .wixproj file for editing

To edit your .wixproj file for editing in Visual Studio:

- 1. Open the project in Visual Studio
- 2. In **Solution Explorer** right click on your project file and select **Unload Project**

Step 2: Add items for prerequisites

Anywhere in your project file, inside the <Project> element, add the following:

These items will be used in step 3 to tell the bootstrap creation task the list of packages to include. In this case the packages are .NET Framework 3.5 and Windows Installer 3.1 (which is a required component for .NET Framework installation).

Step 3: Add the bootstrap generation task

In your project file uncomment the <TargetName="AfterBuild"></Target> element at the end of the file and replace it with the following:

This will instruct MSBuild to generate the bootstrapper after the build of your installer is complete. The ApplicationFile attribute will resolve to the location of your application's installer after the build is complete. The ApplicationName attribute is the application name displayed to the user while the bootstrapper is running. The BootstrapperItems attribute provides the list of pre-requisites to include (from Step 2). The ComponentsLocation attribute is set to Relative to indicate the pre-requisites will be installed from the same location as your application's installer. The CopyComponents attribute is set to true to copy the pre-requisite files into the output directory. The OutputPath attribute resolves to the output location of your installer on disk.

The Path attribute indicates the location on your machine of the prerequisite packages. The location shown above is appropriate for machines with Visual Studio 2008 installed to the default location. For machines with Visual Studio 2005 the default location is C:\Program Files\Microsoft Visual Studio 8\SDK\v2.0\BootStrapper\Packages.

Step 4: Build the project

Do the following to re-open the project for building:

- 1. Save the changes
- 2. In **Solution Explorer** right click on your project file and select **Reload Project**
- 3. In the resulting confirmation dialog select Yes

Then build your project. After your installer is built the bootstrapper will build and be placed in the output directory.

Installing Other Packages

Other packages can be installed using the same mechanism described above. The only additional steps are to modify the list of bootstrapper files in Step 2. The easiest way to obtain the necessary entries is to use the <u>Bootstrapper Manifest Generator</u> tool to create a new MSBuild file with the required packages selected. Then save the generated file, open it in a text editor, and copy out the appropriate entries. The Bootstrap Manifest Generator tool can also be used to create your own custom packages that are then installed via the ClickOnce bootstrapper.

How To: Install DirectX 9.0 With Your Installer

Applications that require components from DirectX 9.0 can benefit from including the DirectX 9.0 Redistributable inside their installer. This simplifies the installation process for end users and ensures the required components for your application are always available on the target user's machine.

DirectX 9.0 can be re-distributed in several different ways, each of which is outlined in MSDN's <u>Installing DirectX with DirectSetup</u> article. This how to describes using the dxsetup.exe application to install DirectX 9.0 on a Vista machine assuming the application being installed only depends on a specific DirectX component.

Prior to redistributing the DirectX binaries you should read and understand the license agreement for the redistributable files. The license agreement can be found in the **Documentation\License**Agreements\DirectX Redist.txt file in your DirectX SDK installation.

Step 1: Add the installer files to your WiX project

Adding the files to the WiX project follows the same process as described in <u>How To: Add a file to your installer</u>. The following example illustrates a typical fragment that includes the necessary files:

```
<DirectoryRef Id="APPLICATIONROOTDIRECTORY">
  <Directory Id="DirectXRedistDirectory" Name="DirectX9.0c">
    <Component Id="DirectXRedist" Guid="PUT-GUID-HERE">
      <File Id="DXSETUPEXE"
     Source="MySourceFiles\DirectXMinInstall\dxsetup.exe"
          KeyPath="yes"
     Checksum="ves"/>
      <File Id="dxupdate.cab"
     Source="MySourceFiles\DirectXMinInstall\dxupdate.cab"/>
      <File Id="dxdllreg x86.cab"
     Source="MySourceFiles\DirectXMinInstall\dxdllreg_x86.cab"/>
      <File Id="dsetup32.dll"
     Source="MySourceFiles\DirectXMinInstall\dsetup32.dll"/>
      <File Id="dsetup.dll"
         Source="MySourceFiles\DirectXMinInstall\dsetup.dll"/>
      <File Id="DEC2006 d3dx9 32 x86.cab"
     Source="MySourceFiles\DirectXMinInstall\DEC2006 d3dx9 32 x86.cab"/>
    </Component>
  </Directory>
</DirectoryRef>
<Feature Id="DirectXRedist"</pre>
    Title="!(loc.FeatureDirectX)"
    AllowAdvertise="no"
Display="hidden" Level="1">
  <ComponentRef Id="DirectXRedist"/>
</Feature>
```

The files included are the minimal set of files required by the DirectX 9.0 install process, as described in the MSDN documentation. The last file in the list, DEC2006_d3dx9_32_x86.cab contains the specific DirectX component required by the installed application. These files are all included in a single component as, even in a patching situation, all the files must go together. A Feature element is used to create a feature specific to DirectX installation, and its Display attribute is set to hidden to

prevent the user from seeing the feature in any UI that may be part of your installer.

Step 2: Add a custom action to invoke the installer

To run the DirectX 9.0 installer a custom action is added that runs before the install is finalized. The CustomAction, InstallExecuteSequence and Custom elements are used to create the custom action, as illustrated in the following sample.

```
<CustomAction Id="InstallDirectX"
    FileKey="DXSETUPEXE"
    ExeCommand="/silent"
    Execute="deferred"
    Impersonate="no"
    Return="check"/>

<InstallExecuteSequence>
    <Custom Action="InstallDirectX" Before="InstallFinalize">
         <![CDATA[NOT REMOVE]]>
         </Custom>
</InstallExecuteSequence>
```

The CustomAction element creates the custom action that runs the setup. It is given a unique id, and the FileKey attribute is used to reference the installer application from Step 1. The ExeCommand attribute adds the *I*silent flag to the installer to ensure the user is not presented with any DirectX installer user interface. The Execute attribute is set to deferred and the Impersonate attribute is set to no to ensure the custom action will run elevated, if necessary. The Return attribute is set to check to ensure the custom action runs synchronously.

The Custom element is used inside an InstallExecuteSequence to add the custom action to the actual installation process. The Action attribute references the CustomAction by its unique id. The Before attribute is set to InstallFinalize to run the custom action before the overall installation is complete. The condition prevents the DirectX installer from running when the user uninstalls your application, since DirectX components cannot be uninstalled.

Step 3: Include progress text for the custom action

If you are using standard WiX UI dialogs you can include custom progress text for display while the DirectX installation takes place. The and elements are used, as illustrated in the following example.

The ProgressText element uses the Action attribute to reference the custom action by its unique id. The value of the ProgressText element is set to the display text for the install progress.

How To: Install the Visual C++ Redistributable with your installer

If your application depends on the Visual C++ runtimes you can include them as part of your installer to simplify the installation experience for your end users. This how to describes including the Visual C++ runtime merge modules into your installer and explains the expected ICE warnings you will see.

Step 1: Obtain the correct Visual C++ runtime merge modules

The Visual C++ runtime merge modules are installed with Visual Studio and are located in \Program Files\Common Files\Merge Modules. The Visual C++ 8.0 runtime file is Microsoft_VC80_CRT_x86.msm. This same MSM is used for the Visual C++ 8.0 SP1 runtime, however it is updated in place by the Visual Studio 2005 SP1 installer. The Visual Studio 9.0 runtime file is Microsoft_VC90_CRT_x86.msm. There is generally no need to include the policy MSMs as part of the installation.

Step 2: Include the merge module in your installer

To include the merge module in your installer use the MergeRef elements. The following example illustrates how these elements are used.

The Merge element ensures the merge module is included in the final Windows Installer package. A unique id is assigned using the Id attribute. The SourceFile attribute points to the location of the merge module on your machine. The Diskld attribute should match the Diskld specified in your project's Media element. The Language attribute should always be 0.

The MergeRef element is used within a Feature element to actually install the merge module. In the example above a feature specific to the runtime is created and marked as hidden to prevent it from displaying in any UI your installer may use. The MergeRef refers to the merge module by its unique id.

A note about ICE warnings

Including the Visual C++ Runtime merge module in your installer will result in the following ICE warnings:

```
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater than length
light.exe(0,0): warning LGHT1076: ICE25: Possible dependency failure as we d
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.98CB24A
```

These warnings are expected and are due to how the Visual C++ merge modules were authored. For more details see <u>Aaron Stebner's blog entry</u>.

How To: Block Installation Based on OS Version

Windows Installer provides the standard <u>VersionNT</u> property that can be used to detect the version of the user's operating system. Often it is desirable to use this property to block installation of an application on incompatible versions of an operating system. The following sample demonstrates how to use this property to block installation of an application on operating systems prior to Windows Vista/Windows Server 2008.

<Condition Message="This application is only supported on Windows Vista, Windows Server 2008, or high
<![CDATA[Installed OR (VersionNT >= 600)]]>
</Condition>

Installed is a Windows Installer property that ensures the check is only done when the user is installing the application, rather than on a repair or remove. The VersionNT part will pass if the property's value is greater than or equal to 600, the version that matches Windows Vista, the installation will proceed. The values for different versions of the Windows operating system are available on MSDN.

To check for versions of 64-bit Windows use the <u>VersionNT64</u> property. To check for versions of Windows prior to Windows NT use the <u>Windows9X</u> property.

How To: Updates

This section includes guides for building updates for your installer.

• How To: Implement a major upgrade in your installer

How To: Implement a Major Upgrade In Your Installer

When creating an .msi-based installer, you are strongly encouraged to include logic that supports <u>Windows Installer major upgrades</u>. Major upgrades are the most common form of updates for .msi's, and including support in your initial .msi release gives you flexibility in the future. Without including support for major upgrades you risk greatly complicating your distribution story if you ever need to release updates later on.

You can use the following steps to enable major upgrades in your .msi, build multiple versions of your .msi and test major upgrade scenarios.

Step 1: Add upgrade information needed to cause new versions to upgrade older versions

In order to allow major upgrades, you must include the following information in your .msi:

Add a unique ID to identify that the product can be upgraded

To accomplish this, you must include an UpgradeCode attribute in your Product element. This looks like the following:

```
<Product Id="*"

UpgradeCode="PUT-GUID-HERE"

Name="My Application Name"

Language="1033"

Version="1.0.1"

Manufacturer="My Manufacturer Name"/>
```

Define the range of old versions that should be upgraded by the new .msi

The <u>UpgradeVersion</u> element will use 3-part version numbers. The minimum version value is typically set to 1.0.0, and the maximum version value is typically set to the 3-part value of the current .msi's version. This looks like the following:

```
<up><upgrade Id="PUT-GUID-HERE"></upgrade Version Minimum="1.0.0"</p>
Include Minimum="yes"
Maximum="1.0.1"
Property="OLDERVERSIONBEINGUPGRADED" />

</Upgrade>
```

The exact name of the property specified in the <u>UpgradeVersion</u> element does not matter, but it must be in all capital letters.

Schedule the removal of old versions of the .msi

There are several options for where you can schedule the RemoveExistingProducts action to remove old versions of the .msi. You need to review the options and choose the one that makes the most sense for your scenarios. You can find a summary of the options in the RemoveExistingProducts documentation. If you choose to schedule it after InstallInitialize, it will look like the following:

```
<InstallExecuteSequence>
    <RemoveExistingProducts After="InstallInitialize"/>
    </InstallExecuteSequence>
```

If you do not schedule the <u>RemoveExistingProducts</u> action, you will see an error like the following:

error LGHT0094: Unresolved reference to symbol 'WixAction:InstallExecuteSe

Windows Installer looks for other installed .msi files with the same UpgradeCode value during the <u>FindRelatedProducts</u> action. If you do not specifically schedule the <u>FindRelatedProducts</u> action in your setup authoring, WiX will automatically schedule it for you when it creates your .msi.

Step 2: Add logic to handle out-of-order installations (installing version 2 then trying to install version 1)

The information provided in step 1 will allow your .msi to uninstall older versions of your .msi during the install process for newer versions. In order to be complete, you should also include information in your .msi to handle scenarios where a user attempts to install a newer version of your .msi and then install an older version afterwards (an out-of-order installation). This step is not strictly necessary, but including this information in your .msi allows you to provide a more user-friendly experience in the case of an out-of-order installation scenario.

Detecting an out-of-order installation requires authoring an <u>UpgradeVersion</u> element that defines a property that will be set if a newer version of the .msi is found on the user's system. This looks like the following:

Once you have defined the detection property, you need to decide how you want your .msi to behave in an out-of-order installation scenario and author an appropriate custom action. There are a couple of options:

Option 1: Block installation

You can block the installation by adding a launch condition that runs if the version detection property is set. This looks like the following:

```
<Condition Message="A later version of [ProductName] is already installed. Setup will now exit.">
NOT NEWERVERSIONDETECTED OR Installed
</condition>
```

Option 2: Immediately exit and return success

This requires creating a custom action that returns exit code 5 (ERROR_NO_MORE_ITEMS). WiX has a built-in custom action named WixExitEarlyWithSuccess that can be used to enable this functionality. To use the built-in custom action, you must make sure that the property created above is named NEWERVERSIONDETECTED. Then, you must reference the custom action by adding the following to your setup authoring:

<CustomActionRef Id="WixExitEarlyWithSuccess"/>

You must also reference the <u>WixUtilExtension</u> to use the WixExitEarlyWithSuccess custom action, either by adding it to the references list for your project if you are using <u>Votive and Visual Studio</u>, or by passing it into <u>light.exe</u> with the -ext command line switch.

An .msi may want to immediately exit and return success instead of blocking and returning an error in an out-of-order installation scenario to support backwards compatibility for calling applications. This is particularly useful if the .msi is a redistributable component that can be shipped and installed as a part of other products.

Step 3: Build version 1 and version 2 of your .msi

Creating version 1 of your .msi is as simple as running your standard build process - this means you compile and link it with the WiX toolset. In order to create version 2 of your .msi, you must make the following changes to your setup authoring, then re-run your build process to create a new .msi:

- Increment the Version value in your <u>Product</u> element to be higher than any previous versions that you have shipped. Windows Installer only uses the first 3 parts of the version in upgrade scenarios, so make sure to increment your version such that one of the first 3 parts is higher than any previously shipped version. For example, if your version 1 uses Version value 1.0.1.0, then version 2 should have a Version value of 1.0.2.0 or higher (1.0.1.1 will not work here).
- Generate a new Id value in the Product element of the new version of the .msi.

Step 4: Test upgrade scenarios before you ship version 1

This step is very important and is too often ignored. In order to make sure that upgrade scenarios will behave the way you expect, you should test upgrades before you ship the first version of your .msi. There are some upgrade-related bugs that can be fixed purely by making fixes in version 2 or higher of your .msi, but there are some bugs that affect the uninstall of version 1 that must be fixed before you ship version 1. Once version 1 ships, you are essentially locked into the uninstall behavior that you ship with version 1, and that impacts major upgrade scenarios because Windows Installer performs an uninstall of version 1 behind the scenes during version 2 installation.

Here are some interesting scenarios to test:

- Install version 1, then install version 2. Make sure that version 1 is correctly removed and version 2 functions correctly. Make sure version 2 cleanly uninstalls afterwards.
- Install version 2, then try to install version 1. Make sure that version 1 correctly detects that version 2 is already installed and either blocks or silently exits, depending on what behavior you choose to implement for your out-of-order installation scenarios.

When testing major upgrade scenarios, make sure to pay particular attention to the conditions on custom actions in your .msi because you may run into issues caused by custom actions running during a major upgrade uninstall and leaving your product in a partially installed state. The UPGRADINGPRODUCTCODE property can be useful to prevent actions from running during an uninstall that is invoked by the RemoveExistingProducts action.

In addition, pay attention to assemblies that need to be installed to the GAC or the Win32 WinSxS store. There is some information about a sequence of events that can remove assemblies from the GAC and the WinSxS store during some major upgrades in this knowledge base article.

How To: General How Tos

This section includes guides to general topics such as debugging and logging installations.

- How To: Get a log of your installation for debugging
- How To: Look inside your MSI with Orca
- How To: Generate a GUID

How To: Get a Log of Your Installation for Debugging

When authoring installers it is often necessary to get a log of the installation for debugging purposes. This is particularly helpful when trying to debug file searches and launch conditions. To obtain a log of an installation use the command line msiexec tool:

msiexec /i MyApplication.msi /l*v MyLogFile.txt

This will install your application and write a verbose log to MyLogFile.txt in the current directory.

If you need to get a log of your installer when it is launched from the Add/Remove Programs dialog you can <u>enable Windows Installer logging</u> <u>via the registry</u>.

How To: Look Inside Your MSI With Orca

When building installers it can often be useful to look inside your installer to see the actual tables and values that were created by the WiX build process. Microsoft provides a tool with the <u>Windows Installer 4.5 SDK</u>, called Orca, that can be used for this purpose. To install Orca, download and install the Windows Installer 4.5 SDK. After the SDK installation is complete navigate to the install directory (typically **C:\Program Files\Windows Installer 4.5 SDK**) and open the **Tools** folder. Inside the Tools folder run Orca.msi to complete the installation.

Once Orca is installed you can right click on any MSI file from Windows Explorer and select **Edit with Orca** to view the contents of the MSI.

How To: Generate a GUID

GUIDs are used extensively with the Windows Installer to uniquely identify products, components, upgrades, and other key elements of the installation process. To generate GUIDs use the <u>guidgen tool</u> that ships with Visual Studio, generally located under **Tools > Create GUID** menu, or the <u>GuidGen.com</u> site. GUIDs generated this way will work fine in WiX, however since they are in mixed case they may cause issues if you share them with users of other, non-WiX tools. For complete compatibility be sure to <u>change the letters in the GUID to uppercase</u> prior to use.

All examples in the How To documentation use the text **PUT-GUID-HERE** for GUIDs. Every **PUT-GUID-HERE** must be replaced with a newlygenerated GUID.

The <<u>Component></u>, <<u>Package></u>, <<u>Patch></u>, <<u>Product></u> elements support auto-generation of GUIDs every time you build your project by specifying a * in place of the GUID. For example:

```
<Product Id="*"
Version="1.0.0.0"
Language="1033"
Name="My Application Name"
Manufacturer="My Manufacturer Name">
```

For the Component element the generated GUID is based on the install directory and filename of the KeyPath for the component. This GUID will stay consistent from build-to-build provided the directory and filename of the KeyPath do not change.

Using WiX extensions

The WiX extensions can be used both on the command line and within the Visual Studio IDE. When you use WiX extensions in the Visual Studio IDE, you can also enable IntelliSense for each WiX extension.

Using WiX extensions on the command line

To use a WiX extension when calling the WiX tools from the command line, use the -ext command line parameter and supply the extension assembly (DLL) needed for your project. Each extension DLL must be passed in via separate -ext parameters. For example:

light.exe MySetup.wixobj

- -ext WixUIExtension
- -ext WixUtilExtension
- -ext "C:\My WiX Extensions\FooExtension.dll"
- -out MySetup.msi

Extension assemblies in the same directory as the WiX tools can be referred to without path or .dll extension. Extension assemblies in other directories must use a complete path name, including .dll extension.

Note: <u>Code Access Security</u> manages the trust levels of assemblies loaded by managed code, including WiX extensions. By default, CAS prevents a WiX tool running on a local machine from loading a WiX extension on a network share.

Using WiX extensions in Visual Studio

To use a WiX extension when building in Visual Studio with the WiX Visual Studio package:

- Right-click on the WiX project in the Visual Studio solution explorer and select Add Reference...
- 2. In the Add WiX Library Reference dialog, click on the Browse tab and browse to the WiX extension DLL that you want to include.
- 3. Click the Add button to add a reference to the chosen extension DLL.
- 4. Browse and add other extension DLLs as needed.

To enable IntelliSense for a WiX extension in the Visual Studio IDE, you need to add an XMLNS declaration to the <Wix> element in your .wxs file. For example, if you want to use the NativeImage functionality in the WixNetFxExtension, the <Wix> element would look like the following:

<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi"
xmlns:netfx="http://schemas.microsoft.com/wix/NetFxExtension">

After adding this, you can add an element named <netfx:NativeImage/> and view IntelliSense for the attributes supported by the NativeImage element.

Optimizing builds

WiX provides two ways of speeding up the creation of cabinets for compressing files:

- Multithreaded cabinet creation.
- Cabinet reuse.

Multithreaded cabinet creation

Light uses multiple threads to build multiple cabinets in a single package. Unfortunately, because the CAB API itself isn't multithreaded, a single cabinet is built with one thread. Light uses multiple threads when there are multiple cabinets, so each cabinet is built on one thread.

By default, Light uses the number of processors/cores in the system as the number of threads to use when creating cabinets. You can override the default using Light's -ct switch or the CabinetCreationThreadCount property in a .wixproj project.

You can use multiple cabinets both externally and embedded in the .msi package (using the Media/@EmbedCab attribute).

Cabinet reuse

If you build setups with files that don't change often, you can generate cabinets for those files once, then reuse them without spending the CPU time to re-build and re-compress them.

There are two Light.exe switches involved in cabinet reuse:

-cc (CabinetCachePath property in .wixproj projects)

The value is the path to use to both write new cabinets and, when - reusecab/ReuseCabinetCache is specified, look for cached cabinets.

-reusecab (ReuseCabinetCache property in .wixproj projects) When -cc/CabinetCachePath is also specified, WiX reuses cabinets that don't need to be rebuilt.

WiX automatically validates that a cached cabinet is still valid by ensuring that:

- The number of files in the cached cabinet matches the number of files being built.
- The names of the files are all identical.
- The order of files is identical.
- The timestamps for all files all identical.

Specifying source files

WiX provides two ways of identifying a setup package's payload - the files that are included in the setup and installed on the user's machine.

- By file name and directory tree.
- By explicit source file.

Compiling, linking, and binding

The WiX toolset models a typical C/C++ compiler is how authored is built, with a compiler that parses the WiX source authoring to object files and a linker that combines the object files into an output. For WiX, the output is an .msi package, .msm merge module, or .wixlib library, which have a third phase: binding payload files into the output. Light.exe includes both the linker and binder.

Though WiX source authoring refers to payload files, the compiler never looks at them; instead, only the binder does, when it creates cabinets containing them or copies them to an uncompressed layout.

You can provide the binder with one or more *base input paths* it uses to look for files. It also looks for files relative to the current working directory. Light.exe's -b switch and the BaseInputPaths .wixproj property let you specify one or more base input paths.

Identifying files by name and directory tree

When you use the File/@Name attribute and don't use the File/@Source attribute, the compiler constructs an implicit path to the file based on the file's parent component directory plus the name you supply. So, for example, given the partial authoring

```
<Directory Id="TARGETDIR">
<Directory Name="foo">
<Directory Name="bar">
<Component>
<File Name="baz.txt" />
```

the binder looks for a file foo\bar\baz.txt in the base input paths.

Overriding implicit payload directories

The <u>FileSource</u> attribute for the <u>Directory</u> and <u>DirectoryRef</u> elements sets a new directory for files in that directory or any child directories. For example, given the partial authoring

```
<Directory Id="TARGETDIR">
<Directory Name="foo" FileSource="build\retail\x86">
<Directory Name="bar">
<Component>
<File Name="baz.txt" />
```

the binder looks for a file *build\retail\x86\bar\baz.txt* in the base input paths.

The <u>FileSource</u> attribute can use preprocessor variables or environment variables. If the value is an absolute path, the binder's base input paths aren't used.

Preferred use

If the build tree serving as your payload source is almost identical to the

tree of your installed image and you have a moderate-to-deep directory tree, using implicit paths will avoid repetition in your authoring.

Source directories

The Directory/@SourceName attribute controls both the name of the directory where Light.exe looks for files and the "source directory" in the .msi package. Unless you also want to control the source directory, just use FileSource.

Identifying payload by source files

The File/@Source attribute is a path to the payload file. It can be an absolute path or relative to any base input path. If File/@Source is present, it takes precedence over the implicit path created by Directory/@Name, Directory/@FileSource, and File/@Name.

If you specify File/@Source, you can omit File/@Name because the compiler automatically sets it to the filename portion of the source path.

Preferred use

If the build tree serving as your payload source is different from the tree of your installed image, using File/@Source makes it easy to pick explicit paths than are different than the .msi package's directory tree. You can use multiple base input paths to shorten the File/@Source paths.

For example, the WiX setup .wixproj project points to the output tree for the x86, x64, and ia64 platforms WiX supports and the WiX source tree. Unique filenames can be referred to with just their filenames; files with the same name across platforms use relative paths.

See the WiX authoring in src\Setup*.wxs for examples.

Standard Custom Actions

The WiX toolset contains several custom actions 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 custom actions are provided in WiX extensions.

To get started using standard custom actions, see the <u>Using Standard Custom Actions</u> topic.

For information about specific types of standard custom actions, see the following topics:

- <u>FileShare custom action</u> (located in WixUtilExtension) create and configure file shares.
- <u>Internet shortcut custom action</u> (located in WixUtilExtension) create shortcuts that point to Web sites.
- OSInfo custom actions (located in WixUtilExtension) set properties for OS information and standard directories that are not provided by default by Windows Installer.
- <u>Performance Counter custom action</u> (located in WixUtilExtension) install and uninstall performance counters.
- Quiet Execution custom action (located in WixUtilExtension) launch console executables without displaying a window.
- <u>Secure Objects custom action</u> (located in WixUtilExtension) secure (using ACLs) objects that the <u>LockPermissions table</u> cannot.
- <u>Service Configuration custom action</u> (located in WixUtilExtension) configure attributes of a Windows service that the <u>ServiceInstall table</u> cannot.
- <u>ShellExecute custom action</u> (located in WixUtilExtension) launch document or URL targets via the Windows shell.
- <u>User custom actions</u> (located in WixUtilExtension) create and configure new users.
- <u>WixDirectXExtension</u> custom action that can be used to check the DirectX capabilities of the video card on the system.
- <u>WixExitEarlyWithSuccess</u> (located in WixUtilExtension) custom action that can be used to exit setup without installing the product.

This can be useful in some major upgrade scenarios.

- <u>WixFailWhenDeferred</u> (located in WixUtilExtension) custom action that can be used to simulate installation failures to test rollback scenarios.
- <u>WixFirewallExtension</u> Firewall custom action that can be used to add exceptions to the Windows Firewall.
- <u>WixGamingExtension</u> Gaming custom action that can be used to add icons and tasks to Windows Game Explorer.
- <u>WixIIsExtension</u> Internet Information Services (IIS) custom actions that can be used to create and configure web sites, virtual directories, web applications, etc.
- <u>WixNetFxExtension</u> custom action to generate native code for .NET assemblies; properties to detect .NET Framework install state and service pack levels.
- <u>WixSqlExtension</u> SQL Server custom actions that can be used to create databases and execute SQL scripts and statements.
- <u>WixVSExtension</u> custom action to register help collections and Visual Studio packages; properties to detect install state and service pack levels for various Visual Studio editions.
- XmlFile custom action (located in WixUtilExtension) configure and modify XML files as part of your installation package.

Using Standard Custom Actions

Custom actions add the ability to install and configure 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. The only things you need to do are understand the appropriate elements for the resources you want to install and set the required attributes on these elements. The elements need to be prefixed with the appropriate namespace for the WiX extension they are defined in. You must pass the full path to the extension DLL as part of the command lines to the compiler and linker so they automatically add the all of the proper error messages, custom action records, and binary records into your final MSI.

Example

First, let's try an example that creates a user account when the MSI is installed. This functionality is defined in WixUtilExtension.dll and exposed to the user as the <User> element.

This is a simple example that will create a new user on the machine named "testName1" with the password "pa\$\$word" (the preprocessor replaces \$\$\$\$ with \$\$).

To build the MSI from this WiX authoring:

- 1. Put the above code in a file named yourfile.wxs.
- 2. Replace the "PUT-GUID-HERE" attributes with real GUIDs.
- 3. Run 'candle.exe yourfile.wxs -ext %full path to WixUtilExtension.dll%'
- 4. Run 'light.exe yourfile.wixobj -ext %full path to WixUtilExtension.dll% –out yourfile.msi yourfile.wixout'

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 added into the MSI. This happened because you have done two key things:

- 1. You made use of a <User> element under a <Component> element. This indicates that a user is to be installed as part of the MSI package, and the WiX compiler automatically added the appropriate MSI table data used by the custom action.
- 2. You linked with the appropriate extension DLL (WixUtilExtension.dll). This caused the linker to automatically pull all of the relevant custom actions, error messages, and binary table rows into the MSI.

OSInfo custom actions

The WixQueryOsInfo, WixQueryOsDirs, and WixQueryOsDriverInfo custom actions in wixca (part of WixUtilExtension) set properties over and above the MSI set for OS product/suite detection and standard directories. The WixQueryOsWellKnownSID custom action sets properties for the localized names of some built in Windows users and groups.

To use these custom actions you simply need to add a <a href="Prop

```
<PropertyRef Id="WIX_SUITE_SINGLEUSERTS" />
<PropertyRef Id="WIX_DIR_COMMON_DOCUMENTS" />
<PropertyRef Id="WIX_ACCOUNT_LOCALSERVICE" />
```

WixUtilExtension will automatically schedule the custom actions as needed after the AppSearch standard action. For additional information about standard directory tokens in Windows and which ones are supported directly by Windows Installer, see the following topics in the MSDN documentation:

- Constant special item ID list (CSIDL) values
- Windows Installer system folder values

WixQueryOsInfo Properties

WIX_SUITE_BACKOFFICE	Equivalent to the OSVEF VER_SUITE_BACKOFF
WIX_SUITE_BLADE	Equivalent to the OSVEF VER_SUITE_BLADE flag
WIX_SUITE_COMMUNICATIONS	Equivalent to the OSVEF VER_SUITE_COMMUNI
WIX_SUITE_COMPUTE_SERVER	Equivalent to the OSVEF VER_SUITE_COMPUTE
WIX_SUITE_DATACENTER	Equivalent to the OSVEF VER_SUITE_DATACEN
WIX_SUITE_EMBEDDEDNT	Equivalent to the OSVEF VER_SUITE_EMBEDDE
WIX_SUITE_EMBEDDED_RESTRICTED	Equivalent to the OSVEF VER_SUITE_EMBEDDE
WIX_SUITE_ENTERPRISE	Equivalent to the OSVEF VER_SUITE_ENTERPR
WIX_SUITE_MEDIACENTER	Equivalent to the GetSys SM_SERVERR2 flag.
WIX_SUITE_PERSONAL	Equivalent to the OSVEF VER_SUITE_PERSONA
WIX_SUITE_SECURITY_APPLIANCE	Equivalent to the OSVEF

	VER_SUITE_SECURITY
WIX_SUITE_SERVERR2	Equivalent to the GetSys SM_SERVERR2 flag.
WIX_SUITE_SINGLEUSERTS	Equivalent to the OSVEF VER_SUITE_SINGLEUS
WIX_SUITE_SMALLBUSINESS	Equivalent to the OSVEF VER_SUITE_SMALLBU
WIX_SUITE_SMALLBUSINESS_RESTRICTED	Equivalent to the OSVEF VER_SUITE_SMALLBU! flag.
WIX_SUITE_STARTER	Equivalent to the GetSys SM_STARTER flag.
WIX_SUITE_STORAGE_SERVER	Equivalent to the OSVEF VER_SUITE_STORAGE
WIX_SUITE_TABLETPC	Equivalent to the GetSys SM_TABLETPC flag.
WIX_SUITE_TERMINAL	Equivalent to the OSVEF VER_SUITE_TERMINAL
WIX_SUITE_WH_SERVER	Windows Home Server. I OSVERSIONINFOEX VER_SUITE_WH_SERV

WixQueryOsDirs Properties

WIX_DIR_ADMINTOOLS	Per-user administrative tools directory. Equivalent to the SHGetFolderPath CSIDL_ADMINTOOLS flag.
WIX_DIR_ALTSTARTUP	Per-user nonlocalized Startup program group. Equivalent to the SHGetFolderPath CSIDL_ALTSTARTUP flag.
WIX_DIR_CDBURN_AREA	Per-user CD burning staging directory. Equivalent to the SHGetFolderPath CSIDL_CDBURN_AREA flag.
WIX_DIR_COMMON_ADMINTOOLS	All-users administrative tools directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_ADMINTOOLS flag.
WIX_DIR_COMMON_ALTSTARTUP	All-users nonlocalized Startup program group. Equivalent to the SHGetFolderPath CSIDL_COMMON_ALTSTARTUP flag.
WIX_DIR_COMMON_DOCUMENTS	All-users documents directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_DOCUMENTS flag.

WIX_DIR_COMMON_FAVORITES	All-users favorite items directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_FAVORITES flag.
WIX_DIR_COMMON_MUSIC	All-users music files directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_MUSIC flag.
WIX_DIR_COMMON_PICTURES	All-users picture files directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_PICTURES flag.
WIX_DIR_COMMON_VIDEO	All-users video files directory. Equivalent to the SHGetFolderPath CSIDL_COMMON_VIDEO flag.
WIX_DIR_COOKIES	Per-user Internet Explorer cookies directory. Equivalent to the SHGetFolderPath CSIDL_COOKIES flag.
WIX_DIR_DESKTOP	Per-user desktop directory. Equivalent to the SHGetFolderPath CSIDL_DESKTOP flag.
WIX_DIR_HISTORY	Per-user Internet Explorer history directory. Equivalent to the SHGetFolderPath CSIDL_HISTORY flag.

WIX_DIR_INTERNET_CACHE	Per-user Internet Explorer cache directory. Equivalent to the SHGetFolderPath CSIDL_INTERNET_CACHE flag.
WIX_DIR_MYMUSIC	Per-user music files directory. Equivalent to the SHGetFolderPath CSIDL_MYMUSIC flag.
WIX_DIR_MYPICTURES	Per-user picture files directory. Equivalent to the SHGetFolderPath CSIDL_MYPICTURES flag.
WIX_DIR_MYVIDEO	Per-user video files directory. Equivalent to the SHGetFolderPath CSIDL_MYVIDEO flag.
WIX_DIR_NETHOOD	Per-user My Network Places link object directory. Equivalent to the SHGetFolderPath CSIDL_NETHOOD flag.
WIX_DIR_PERSONAL	Per-user documents directory. Equivalent to the SHGetFolderPath CSIDL_PERSONAL flag.
WIX_DIR_PRINTHOOD	Per-user Printers link object directory. Equivalent to the SHGetFolderPath CSIDL_PRINTHOOD flag.
WIX_DIR_PROFILE	Per-user profile directory. Equivalent to the

	SHGetFolderPath CSIDL_PROFILE flag.
WIX_DIR_RECENT	Per-user most recently used documents shortcut directory. Equivalent to the SHGetFolderPath CSIDL_RECENT flag.
WIX_DIR_RESOURCES	All-users resource data directory. Equivalent to the SHGetFolderPath CSIDL_RESOURCES flag.

WixQueryOsWellKnownSID properties

WIX_ACCOUNT_LOCALSYSTEM	Localized qualified name of the Local System account.
WIX_ACCOUNT_LOCALSERVICE	Localized qualified name of the Local Service account.
WIX_ACCOUNT_NETWORKSERVICE	Localized qualified name of the Network Service account.
WIX_ACCOUNT_ADMINISTRATORS	Localized qualified name of the Administrators group.
WIX_ACCOUNT_USERS	Localized qualified name of the Users group.
WIX_ACCOUNT_GUESTS	Localized qualified name of the Users group.

WixQueryOsDriverInfo properties

WIX_WDDM_DRIVER_PRESENT Set to 1 if the video card driver

on the target machine is a WDDM driver. This property is only set on machines running

Windows Vista or higher.

WIX_DWM_COMPOSITION_ENABLED Set to 1 if the target machine

has composition enabled. This

property is only set on machines running Windows

Vista or higher.

Performance Counter Custom Action

The PerfCounter element (part of WiXUtilExtension) 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. <u>This MSDN</u> <u>documentation</u> is a good place to start. See below for samples repurposed from MSDN.
- The RegisterPerfmon custom action You can link with the WiXUtilExtension.dll to ensure that the custom actions are included in your final MSI. See <u>Using Standard Custom Actions</u>. 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/2006/wi">
<Fragment>
  <DirectoryRef Id="BinDir">
   <Component Id="SharedNative" DiskId="1">
    <Registry Id="Shared_r1" Root="HKLM" Key="SYSTEM\CurrentControl
    <Registry Id="Shared_r2" Root="HKLM" Key="SYSTEM\CurrentControl</pre>
    <Registry Id="Shared_r3" Root="HKLM" Key="SYSTEM\CurrentControl
    <Registry Id="Shared_r4" Root="HKLM" Key="SYSTEM\CurrentControl
   <File Id="PERFDLL.DLL" Name="MyPerfDll.dll" Source="x86\debug\0\n</pre>
   <File Id="PERFCOUNTERS.H" Name="PerfCounters.h" Source="x86\deb"
   <File Id="PERFCOUNTERS.INI" Name="PerfCounters.ini" Source="x86\/
     <PerfCounter Name="MyApplication" />
   </File>
   </Component>
  </DirectoryRef>
</Fragment>
</Wix>
```

```
Sample PerfCounters.ini:
[info]
drivername=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 Chines
DEVICE_COUNTER_1_009_NAME=Name of first counter
DEVICE_COUNTER_1_009_HELP=Displays the current value of the first cou
DEVICE COUNTER 1 004 NAME=Name of the first counter in Chinese
DEVICE_COUNTER_1_004_HELP=Displays the value of the first counter in C
DEVICE COUNTER 2 009 NAME=Name of the second counter
DEVICE_COUNTER_2_009_HELP=Displays the current rate of the second cou
DEVICE_COUNTER_2_004_NAME=Name of the second counter in Chinese
DEVICE_COUNTER_2_004_HELP=Displays the rate of the second counter in
PERF_OBJECT_1_009_NAME=Name of the third counter
PERF_OBJECT_1_009_HELP=Displays the current rate of the third counter
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 Chinese
Sample PerfCounters.h:
#define OBJECT 1
#define DEVICE COUNTER 1
#define DEVICE COUNTER 2
#define PERF OBJECT 1 8
```

Quiet Execution Custom Action

The QtExec custom action allows you to run an arbitrary command line in an MSI-based setup in silent mode. QtExec is commonly used to suppress console windows that would otherwise appear appear when invoking the executable directly. The custom action is located in the WixCA library, which is a part of the WixUtilExtension.

Immediate execution

When the QtExec action is run as an immediate custom action, it will try to execute the command stored in the QtExecCmdLine property. The following is an example of authoring an immediate QtExec custom action:

This will result in running the command line in the immediate sequence. If the exit code of the command line in this example indicates an error (meaning that the return code is not equal to 0) then the setup will fail because the Return value is set to "check." Changing the Return value to "ignore" will cause the setup to log the failure but skip it and continue instead of failing the entire setup.

If you want to run more than one command line in the immediate sequence then you will need to schedule multiple QtExec custom actions and set the QtExecCmdLine property to a new value by scheduling a property-setting custom action immediately before each instance of the QtExec custom action.

Deferred execution

When the QtExec action is run as a deferred custom action, it will try to execute the command line stored in the value of the custom action data. For deferred QtExec custom actions, the custom action data is a property that has the same Id value as the custom action Id. The following is an example of authoring a deferred QtExec custom action:

If you need to set a command line that uses other Windows Installer properties, you must schedule an immediate custom action to set the command line property value and schedule a deferred custom action to run QtExec. The Property value used in the immediate custom action must match the Id value used in the deferred custom action. A common use of this pattern for QtExec custom actions is to run an executable that will be installed as a part of the setup. The following is an example of authoring a deferred QtExec custom action that relies on another property value:

Running 64-bit executables

If you need to run a 64-bit executable, use the 64-bit aware QtExec. To use the 64-bit QtExec change the CustomAction element's DllEntry attribute to "CAQuietExec64" and for immediate execution use the "QtExec64CmdLine" property. The following example combines the examples above the 64-bit aware QtExec for both. Notice that the CustomAction element's Id attributes do not need to change:

Building an MSI that uses QtExec

In order to use QtExec, you must include a reference to the WixUtilExtension when building your MSI. To do this, add the command line argument -ext WixUtilExtension.dll when calling Light.exe.

ShellExecute CustomAction

The WixShellExec custom action in wixca (part of WixUtilExtension) lets you open document or URL targets via the Windows shell. A common use is to launch readme files or URLs using their registered default applications based on their extension. Note that WixShellExecute can only be used as an immediate custom action as it launches an application without waiting for it to close. WixShellExec reads its target from the WixShellExecTarget property, formats it, and then calls ShellExecute with the formatted value. It uses the default verb, which is usually "open." For more information, see ShellExecute Function.

For a step-by-step example of how to use the ShellExecute custom action to launch a program at the end of install see the <u>How To: Run the Installed Application After Setup</u> topic.

WixDirectXExtension

The WixDirectXExtension includes a custom action named WixQueryDirectXCaps that sets properties you can use to check the DirectX capabilities of the installing user's video card.

WixDirectXExtension properties

WIX_DIRECTX_PIXELSHADERVERSION	Pixel shader version capabi major*100 + minor. For exa model 3.0-compliant system WIX_DIRECTX_PIXELSHA value of 300.

WIX_DIRECTX_VERTEXSHADERVERSION Vertex shader version capa as major*100 + minor. For e shader model 3.0-complian have a WIX_DIRECTX_VERTEXS value of 300.

To use the WixDirectXExtension properties in an MSI, use the following steps:

- Add PropertyRef elements for items listed above that you want to use in your MSI.
- Add the -ext <path to WixDirectXExtension.dll> command line parameter when calling light.exe to include the WixDirectXExtension in the MSI linking process.
- Or, using an MSBuild-based .wixproj project, add <path to WixDirectXExtension.dll> to the WixExtension item group. When using Votive in Visual Studio, this can be done by right-clicking on the References node in a WiX project, choosing Add Reference... then browsing for WixDirectXExtension.dll and adding a reference.

For example:

Note that the WixDirectXExtension properties are set to the value **NotSet** by default. The WixDirectXExtension custom action is configured to not fail if it encounters any errors when trying to determine DirectX capabilities. In this type of scenario, the properties will be set to their **NotSet** default values. In your setup authoring, you can compare the property values to the **NotSet** value or to a specific value to determine whether WixDirectXExtension was able to query DirectX capabilities and if so, what they are.

WixExitEarlyWithSuccess Custom Action

The WixExitEarlyWithSuccess custom action is an immediate custom action that does nothing except return the value ERROR_NO_MORE_ITEMS. This return value causes Windows Installer to skip all remaining actions in the .msi and return a process exit code that indicates a successful installation.

This custom action is useful in cases where you want setup to exit without actually installing anything, but want it to return success to the calling process. A common scenario where this type of behavior is useful is in an out-of-order installation scenario for an .msi that implements major upgrades. When a user has version 2 of an .msi installed and then attempts to install version 1, this custom action can be used in conjunction with the Upgrade table to detect that version 2 is already installed to cause setup to exit without installing anything and return success. If any applications redistribute version 1 of the .msi, their installation processes will continue to work even if the user has version 2 of the .msi installed on their system.

There are 3 steps you need to take to use the WixExitEarlyWithSuccess custom action in your MSI:

Step 1: Add the WiX utilities extensions library to your project

The WiX support for WixExitEarlyWithSuccess is included in a WiX extension library that must be added to your project prior to use. If you are using WiX on the command line you need to add the following to your light command line:

light.exe myproject.wixobj -ext WixUtilExtension

If you are using Votive you can add the extension using the Add Reference dialog:

- 1. Open your Votive project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the WixUtilExtension.dll assembly from the list and click Add
- 4. Close the Add Reference dialog

Step 2: Add a reference to the WixExitEarlyWithSuccess custom action

To add a reference to the WixExitEarlyWithSuccess custom action, include the following in your WiX setup authoring:

<CustomActionRef Id="WixExitEarlyWithSuccess" />

This will cause WiX to add the WixExitEarlyWithSuccess custom action to your MSI, schedule it immediately after the FindRelatedProducts action and condition it to only run if the property named NEWERVERSIONDETECTED is set.

Step 3: Add logic to define the NEWERVERSIONDETECTED property at the appropriate times

In order to cause the WixExitEarlyWithSuccess to run at the desired times, you must add logic to your installer to create the NEWERVERSIONDETECTED property. To implement the major upgrade example described above, you can add an Upgrade element like the following:

```
<Upgrade Id="!(loc.Property_UpgradeCode)">
  <UpgradeVersion Minimum="$(var.ProductVersion)" OnlyDetect="yes" Prope
</Upgrade>
```

WixFailWhenDeferred Custom Action

When authoring <u>deferred custom actions</u> (which are custom actions that change the system state) in an MSI, it is necessary to also provide an equivalent set of rollback custom actions to undo the system state change in case the MSI fails and rolls back. The rollback behavior typically needs to behave differently depending on if the MSI is currently being installed, repaired or uninstalled. This means that the following scenarios need to be accounted for when coding and testing a set of deferred custom actions to make sure that they are working as expected during both success and failure cases:

- 1. Successful install
- 2. Failed install
- 3. Successful repair
- 4. Failed repair
- 5. Successful uninstall
- Failed uninstall

The failure cases are often difficult to simulate by unit testing the custom action code directly because deferred custom action code typically depends on state information provided to it by Windows Installer during an active installation session. As a result, this type of testing usually requires fault injection in order to cause the rollback custom actions to be executed at the proper times during real installation scenarios.

WiX includes a simple deferred custom action named WixFailWhenDeferred to help make it easier to test rollback custom actions in an MSI. WixFailWhenDeferred will always fail when it is executed during the installation, repair or uninstallation of an MSI. Adding the WixFailWhenDeferred custom action to your MSI allows you to easily inject a failure into your MSI in order to test your rollback custom actions.

There are 3 steps you need to take to use the WixFailWhenDeferred custom action to test the rollback custom actions in your MSI:

Step 1: Add the WiX utilities extensions library to your project

The WiX support for WixFailWhenDeferred is included in a WiX extension library that must be added to your project prior to use. If you are using WiX on the command line you need to add the following to your light command line:

light.exe myproject.wixobj -ext WixUtilExtension

If you are using Votive you can add the extension using the Add Reference dialog:

- 1. Open your Votive project in Visual Studio
- 2. Right click on your project in Solution Explorer and select Add Reference...
- 3. Select the WixUtilExtension.dll assembly from the list and click Add
- 4. Close the Add Reference dialog

Step 2: Add a reference to the WixFailWhenDeferred custom action

To add a reference to the WixFailWhenDeferred custom action, include the following in your WiX setup authoring:

<CustomActionRef Id="WixFailWhenDeferred" />

This will cause WiX to add the WixFailWhenDeferred custom action to your MSI, schedule it immediately before the <u>InstallFinalize</u> action and condition it to only run if the property WIXFAILWHENDEFERRED=1.

Step 3: Build your MSI and test various scenarios

The WixFailWhenDeferred custom action is conditioned to run only when the Windows Installer public property WIXFAILWHENDEFERRED=1. After building your MSI with a reference to the WixFailWhenDeferred custom action, you can use the following set of command lines to simulate a series of standard install and rollback testing scenarios:

- Standard install: msiexec.exe /i MyProduct.msi /qb /l*vx %temp%\MyProductInstall.log
- 2. **Install rollback:** msiexec.exe /i MyProduct.msi /qb /l*vx %temp%\MyProductInstallFailure.log WIXFAILWHENDEFERRED=1
- 3. **Standard repair:** msiexec.exe /fvecmus MyProduct.msi /qb /l*vx %temp%\MyProductRepair.log
- 4. **Repair rollback:** msiexec.exe /fvecmus MyProduct.msi /qb /l*vx %temp%\MyProductRepairFailure.log WIXFAILWHENDEFERRED=1
- 5. **Standard uninstall:** msiexec.exe /x MyProduct.msi /qb /l*vx %temp%\MyProductUninstall.log
- Uninstall rollback: msiexec.exe /x MyProduct.msi /qb /l*vx %temp%\MyProductUninstallFailure.log WIXFAILWHENDEFERRED=1

WixGamingExtension

The <u>WixGamingExtension</u> lets you register your application as a game in Windows Vista and later, in three main categories:

- Game Explorer integration with game definition file
- Game Explorer tasks
- Rich saved-game preview

Game Explorer integration

For an overview of Game Explorer, see <u>Getting Started With Game</u> <u>Explorer</u>. Game Explorer relies on an embedded file (game definition file or GDF) to control the data displayed about the game. For details about GDFs, see <u>The Game-Definition-File (GDF) Schema</u> and <u>GDF Delivery and Localization</u>. Using WixGamingExtension, you register a game with Game Explorer using the Game element as a child of your game executable's File element:

```
<File Id="MyGameExeFile" Name="passenger_simulator.exe" KeyPath="yes">
    <gaming:Game Id="985D5FD3-FC40-4CE9-9EE5-F2AAAB959230">
        ...
    </File>
```

The Game/@Id attribute is used as the InstanceID attribute discussed <a href="https://here.ncbi.nlm.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.new.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.neg.go.

Implementation note: Using the Game element adds a row to a custom table in your .msi package and schedules the Gaming custom action; at install time, that custom action adds/updates/removes the game in Game Explorer and for operating system upgrades. (See <u>Supporting an Upgrade from Windows XP to Windows Vista</u> for details.)

Game Explorer tasks

In Game Explorer, a game's context menu includes custom *tasks*:

- Play tasks start the game with optional arguments.
- Support tasks start the user's default browser to go to a specific URL.

For details, see <u>Game Explorer Tasks</u>. In WixGameExtension, PlayTask and SupportTask are child elements of the Game element:

For details, see the **Gaming schema documentation**.

Implementation note: Game Explorer tasks are shortcuts, so the Gaming compiler extension translates the PlayTask into rows in <u>Shortcuts</u> and SupportTask into WixUtilExtension <u>InternetShortcuts</u>. It also creates directories to hold the shortcuts and custom actions to set the directories

Rich saved-game preview

Windows Vista includes a shell handler that lets games expose metadata in their saved-game files. For details, see <u>Rich Saved Games</u>. If your game supports rich saved games, you can register it for the rich saved-game preview using the WixGamingExtension IsRichSavedGame attribute on the <u>Extension element</u>:

```
<ProgId Id="MyGameProgId">
  <Extension Id="MyGameSave" gaming:IsRichSavedGame="yes" />
  </ProgId>
```

Implementation note: The Gaming compiler extension translates the IsRichSavedGame attribute to rows in the MSI Registry table.

WixNetfxExtension

The <u>WixNetfxExtension</u> includes a set of custom actions to compile native images using Ngen.exe. For an example, see <u>How To: NGen managed assemblies during installation</u>.

The WixNetfxExtension also includes a set of properties that can be used to detect the presence of various versions of the .NET Framework, the .NET Framework SDK and the Windows SDK. For information on how to use these properties to verify the user's .NET Framework version at install time see How To: Check for .NET Framework Versions.

Properties

The following property is applicable to all versions of the .NET Framework:

Property name	Meaning
NETFRAMEWORKINSTALLROOTDIR	Set to the root installation director all versions of the .NET Framewo (%windir%\Microsoft.NET\Frame\

Here is a complete list of properties for the **.NET Framework 1.0** product family:

Property name	Meaning
	Set to 3321-3705 if the .NET Finstalled (not set otherwise).
	Set to the installation directory Framework 1.0 (%windir%\Microsoft.NET\Fran

Here is a complete list of properties for the **.NET Framework 1.1** product family:

Property name	Meaning
NETFRAMEWORK11	Set to #1 if the .NET Frame(not set otherwise).
NETFRAMEWORK11_SP_LEVEL	Indicates the service pack le Framework 1.1.

NETFRAMEWORK11INSTALLROOTDIR	Set to the installation director Framework 1.1 (%windir%\Microsoft.NET\F
NETFRAMEWORK11_ZH_CN_LANGPACK	Set to #1 if the .NET Frame((Simplified) language pack is otherwise).
NETFRAMEWORK11_ZH_TW_LANGPACK	Set to #1 if the .NET Frame\ (Traditional) language pack otherwise).
NETFRAMEWORK11_CS_CZ_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_DA_DK_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_NL_NL_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_FI_FI_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_FR_FR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_DE_DE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_EL_GR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_HU_HU_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r

K	Y.
NETFRAMEWORK11_IT_IT_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_JA_JP_LANGPACK	Set to #1 if the .NET Framev language pack is installed (r
NETFRAMEWORK11_KO_KR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_NB_NO_LANGPACK	Set to #1 if the .NET Framev language pack is installed (r
NETFRAMEWORK11_PL_PL_LANGPACK	Set to #1 if the .NET Framev language pack is installed (r
NETFRAMEWORK11_PT_BR_LANGPACK	Set to #1 if the .NET Framev (Brazil) language pack is ins otherwise).
NETFRAMEWORK11_PT_PT_LANGPACK	Set to #1 if the .NET Framev (Portugal) language pack is otherwise).
NETFRAMEWORK11_RU_RU_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK11_ES_ES_LANGPACK	Set to #1 if the .NET Framev language pack is installed (r
NETFRAMEWORK11_SV_SE_LANGPACK	Set to #1 if the .NET Framev language pack is installed (r
NETFRAMEWORK11_TR_TR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
·	· · · · · · · · · · · · · · · · · · ·

Here is a complete list of properties for the **.NET Framework 2.0** product family:

Property name	Meaning
NETFRAMEWORK20	Set to #1 if the .NET Frame set otherwise).
NETFRAMEWORK20_SP_LEVEL	Indicates the service pack le Framework 2.0.
NETFRAMEWORK20INSTALLROOTDIR	Set to the installation director Framework 2.0 (%windir%\Microsoft.NET\F
NETFRAMEWORK20INSTALLROOTDIR64	Set to the installation director Framework 2.0 (%windir%\Microsoft.NET\F
NETFRAMEWORK20_ZH_CN_LANGPACK	Set to #1 if the .NET Frame (Simplified) language pack i otherwise).
NETFRAMEWORK20_ZH_TW_LANGPACK	Set to #1 if the .NET Frame (Traditional) language pack otherwise).
NETFRAMEWORK20_CS_CZ_LANGPACK	Set to #1 if the .NET Frame pack is installed (not set oth
NETFRAMEWORK20_DA_DK_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_NL_NL_LANGPACK	Set to #1 if the .NET Frame pack is installed (not set oth
II I	I

NETFRAMEWORK20_FI_FI_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_FR_FR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_DE_DE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_EL_GR_LANGPACK	Set to #1 if the .NET Frame pack is installed (not set oth
NETFRAMEWORK20_HU_HU_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_IT_IT_LANGPACK	Set to #1 if the .NET Frame pack is installed (not set oth
NETFRAMEWORK20_JA_JP_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_KO_KR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_NB_NO_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_PL_PL_LANGPACK	Set to #1 if the .NET Frame pack is installed (not set oth
NETFRAMEWORK20_PT_BR_LANGPACK	Set to #1 if the .NET Frame (Brazil) language pack is ins otherwise).
NETFRAMEWORK20_PT_PT_LANGPACK	Set to #1 if the .NET Frame\

	(Portugal) language pack is otherwise).
NETFRAMEWORK20_RU_RU_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_ES_ES_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK20_SV_SE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK20_TR_TR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r

Here is a complete list of properties for the .NET Framework 3.0 product family:

Property name	Meaning
NETFRAMEWORK30	Set to #1 if the .NET Frame installed (not set otherwise).
NETFRAMEWORK30_SP_LEVEL	Indicates the service pack le Framework 3.0. This value v service pack is installed.
NETFRAMEWORK30INSTALLROOTDIR	Set to the installation director Framework 3.0 (%windir%\Microsoft.NET\F
NETFRAMEWORK30INSTALLROOTDIR64	Set to the installation director. NET Framework 3.0 (%windir%\Microsoft.NET\F

NETFRAMEWORK30_ZH_CN_LANGPACK	Set to #1 if the .NET Frame (Simplified) language pack i otherwise).
NETFRAMEWORK30_ZH_TW_LANGPACK	Set to #1 if the .NET Frame (Traditional) language pack set otherwise).
NETFRAMEWORK30_CS_CZ_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_DA_DK_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_NL_NL_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_FI_FI_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_FR_FR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_DE_DE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_EL_GR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_HU_HU_LANGPACK	Set to #1 if the .NET Frame Hungarian language pack is otherwise).
NETFRAMEWORK30_IT_IT_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r

NETFRAMEWORK30_JA_JP_LANGPACK	Set to #1 if the .NET Frame Japanese language pack is otherwise).
NETFRAMEWORK30_KO_KR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_NB_NO_LANGPACK	Set to #1 if the .NET Frame Norwegian language pack is otherwise).
NETFRAMEWORK30_PL_PL_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_PT_BR_LANGPACK	Set to #1 if the .NET Frame Portuguese (Brazil) languag (not set otherwise).
NETFRAMEWORK30_PT_PT_LANGPACK	Set to #1 if the .NET Frame Portuguese (Portugal) languinstalled (not set otherwise).
NETFRAMEWORK30_RU_RU_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_ES_ES_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_SV_SE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK30_TR_TR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r

Here is a complete list of properties for the .NET Framework 3.5 product

family:

Property name	Meaning
NETFRAMEWORK35	Set to #1 if the .NET Framevinstalled (not set otherwise).
NETFRAMEWORK35_SP_LEVEL	Indicates the service pack le Framework 3.5.
NETFRAMEWORK35INSTALLROOTDIR	Set to the installation director Framework 3.5 (%windir%\Microsoft.NET\F
NETFRAMEWORK35INSTALLROOTDIR64	Set to the installation director. NET Framework 3.5 (%windir%\Microsoft.NET\F
NETFRAMEWORK35_ZH_CN_LANGPACK	Set to #1 if the .NET Frame (Simplified) language pack is otherwise).
NETFRAMEWORK35_ZH_TW_LANGPACK	Set to #1 if the .NET Frame (Traditional) language pack set otherwise).
NETFRAMEWORK35_CS_CZ_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_DA_DK_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_NL_NL_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r

NETFRAMEWORK35_FI_FI_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_FR_FR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_DE_DE_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_EL_GR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_HU_HU_LANGPACK	Set to #1 if the .NET Frame Hungarian language pack is otherwise).
NETFRAMEWORK35_IT_IT_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_JA_JP_LANGPACK	Set to #1 if the .NET Frame Japanese language pack is otherwise).
NETFRAMEWORK35_KO_KR_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_NB_NO_LANGPACK	Set to #1 if the .NET Frame Norwegian language pack is otherwise).
NETFRAMEWORK35_PL_PL_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_PT_BR_LANGPACK	Set to #1 if the .NET Frame Portuguese (Brazil) languag (not set otherwise).

NETFRAMEWORK35_PT_PT_LANGPACK	Set to #1 if the .NET Frame Portuguese (Portugal) languinstalled (not set otherwise).
NETFRAMEWORK35_RU_RU_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_ES_ES_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_SV_SE_LANGPACK	Set to #1 if the .NET Frame language pack is installed (r
NETFRAMEWORK35_TR_TR_LANGPACK	Set to #1 if the .NET Frame\ language pack is installed (r
NETFRAMEWORK35_CLIENT	Set to #1 if the .NET Frame profile is installed (not set ot
NETFRAMEWORK35_CLIENT_SP_LEVEL	Indicates the service pack le Framework 3.5 client profile

Here is a complete list of properties for the **.NET Framework 4.0** product family:

Property name	Meaning
	Set to #1 if the .NET installed (not set oth available starting wi
	Indicates the service Framework 4.0 full. starting with WiX v3

NETFRAMEWORK40FULLINSTALLROOTDIR	Set to the installatio Framework 4.0 full (%windir%\Microsof This property is ava v3.5.
NETFRAMEWORK40FULLINSTALLROOTDIR64	Set to the installatio .NET Framework 4. (%windir%\Microsof This property is ava v3.5.
NETFRAMEWORK40FULL_AR_SA_LANGPACK	Set to #1 if the .NET Arabic language par otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_ZH_CN_LANGPACK	Set to #1 if the .NET Chinese (Simplified) installed (not set oth available starting wi
NETFRAMEWORK40FULL_ZH_TW_LANGPACK	Set to #1 if the .NET Chinese (Traditional installed (not set oth available starting wi
NETFRAMEWORK40FULL_CS_CZ_LANGPACK	Set to #1 if the .NET Czech language par otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_DA_DK_LANGPACK	Set to #1 if the .NET Danish language pa otherwise). This pro with WiX v3.5.

NETFRAMEWORK40FULL_NL_NL_LANGPACK	Set to #1 if the .NET language pack is ins This property is ava v3.5.
NETFRAMEWORK40FULL_FI_FI_LANGPACK	Set to #1 if the .NET Finnish language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_FR_FR_LANGPACK	Set to #1 if the .NET French language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_DE_DE_LANGPACK	Set to #1 if the .NET German language p otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_EL_GR_LANGPACK	Set to #1 if the .NET language pack is in: This property is ava v3.5.
NETFRAMEWORK40FULL_HE_IL_LANGPACK	Set to #1 if the .NET Hebrew language potherwise). This prowith WiX v3.5.
NETFRAMEWORK40FULL_HU_HU_LANGPACK	Set to #1 if the .NET Hungarian language otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_IT_IT_LANGPACK	Set to #1 if the .NET

	language pack is ins This property is ava v3.5.
NETFRAMEWORK40FULL_JA_JP_LANGPACK	Set to #1 if the .NET Japanese language otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_KO_KR_LANGPACK	Set to #1 if the .NET Korean language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_NB_NO_LANGPACK	Set to #1 if the .NET Norwegian language otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_PL_PL_LANGPACK	Set to #1 if the .NET language pack is ins This property is ava v3.5.
NETFRAMEWORK40FULL_PT_BR_LANGPACK	Set to #1 if the .NET Portuguese (Brazil) (not set otherwise). starting with WiX v3
NETFRAMEWORK40FULL_PT_PT_LANGPACK	Set to #1 if the .NET Portuguese (Portuginstalled (not set oth available starting wi
NETFRAMEWORK40FULL_RU_RU_LANGPACK	Set to #1 if the .NET Russian language p otherwise). This pro

	with WiX v3.5.
NETFRAMEWORK40FULL_ES_ES_LANGPACK	Set to #1 if the .NET Spanish language p otherwise). This pro with WiX v3.5.
NETFRAMEWORK40FULL_SV_SE_LANGPACK	Set to #1 if the .NET Swedish language potherwise). This prowith WiX v3.5.
NETFRAMEWORK40FULL_TR_TR_LANGPACK	Set to #1 if the .NET Turkish language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT	Set to #1 if the .NET profile is installed (n property is available
NETFRAMEWORK40CLIENT_SERVICING_LEVEL	Indicates the service Framework 4.0 clier available starting wi
NETFRAMEWORK40CLIENTINSTALLROOTDIR	Set to the installatio Framework 4.0 full (%windir%\Microsof This property is ava v3.5.
NETFRAMEWORK40CLIENTINSTALLROOTDIR64	Set to the installatio .NET Framework 4.0 (%windir%\Microsof This property is ava v3.5.

NETFRAMEWORK40CLIENT_AR_SA_LANGPACK	Set to #1 if the .NET Arabic language par otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_ZH_CN_LANGPACK	Set to #1 if the .NET Chinese (Simplified) installed (not set oth available starting wi
NETFRAMEWORK40CLIENT_ZH_TW_LANGPACK	Set to #1 if the .NET Chinese (Traditional installed (not set oth available starting wi
NETFRAMEWORK40CLIENT_CS_CZ_LANGPACK	Set to #1 if the .NET Czech language par otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_DA_DK_LANGPACK	Set to #1 if the .NET Danish language pa otherwise). This prowith WiX v3.5.
NETFRAMEWORK40CLIENT_NL_NL_LANGPACK	Set to #1 if the .NET Dutch language pac otherwise). This prowith WiX v3.5.
NETFRAMEWORK40CLIENT_FI_FI_LANGPACK	Set to #1 if the .NET Finnish language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_FR_FR_LANGPACK	Set to #1 if the .NET

	French language pa otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_DE_DE_LANGPACK	Set to #1 if the .NET German language p otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_EL_GR_LANGPACK	Set to #1 if the .NET Greek language pac otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_HE_IL_LANGPACK	Set to #1 if the .NET Hebrew language po otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_HU_HU_LANGPACK	Set to #1 if the .NET Hungarian language otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_IT_IT_LANGPACK	Set to #1 if the .NET Italian language pac otherwise). This prowith WiX v3.5.
NETFRAMEWORK40CLIENT_JA_JP_LANGPACK	Set to #1 if the .NET Japanese language otherwise). This prowith WiX v3.5.
NETFRAMEWORK40CLIENT_KO_KR_LANGPACK	Set to #1 if the .NET Korean language pa otherwise). This pro

	with WiX v3.5.
NETFRAMEWORK40CLIENT_NB_NO_LANGPACK	Set to #1 if the .NET Norwegian language otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_PL_PL_LANGPACK	Set to #1 if the .NET Polish language pac otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_PT_BR_LANGPACK	Set to #1 if the .NET Portuguese (Brazil) (not set otherwise). starting with WiX v3
NETFRAMEWORK40CLIENT_PT_PT_LANGPACK	Set to #1 if the .NET Portuguese (Portuga installed (not set oth available starting wi
NETFRAMEWORK40CLIENT_RU_RU_LANGPACK	Set to #1 if the .NET Russian language p otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_ES_ES_LANGPACK	Set to #1 if the .NET Spanish language p otherwise). This pro with WiX v3.5.
NETFRAMEWORK40CLIENT_SV_SE_LANGPACK	Set to #1 if the .NET Swedish language potherwise). This prowith WiX v3.5.

	NETFRAMEWORK40CLIENT_TR_TR_LANGPACK	Set to #1 if the .NET
ı		Turkish language pa
ı		otherwise). This pro
ı		with WiX v3.5.
ı		

Here is a complete list of properties for the .NET Framework SDK and Windows SDK:

Property name	Meaning
NETFRAMEWORK11SDKDIR	The location of the .NET Framework 1.1 SDK installation root.
NETFRAMEWORK20SDKDIR	The location of the .NET Framework 2.0 SDK installation root.
WINDOWSSDKCURRENTVERSIONDIR	The location of the currently active version of the Windows SDK.
WINDOWSSDKCURRENTVERSION	The version number of the currently active version of the Windows SDK.
WINDOWSSDK60ADIR	The location of the Windows SDK 6.0a installation root.
WINDOWSSDK61DIR	The location of the Windows SDK 6.1 installation root.
WINDOWSSDK70ADIR	The location of the Windows SDK 7.0a installation root. This property is available starting with WiX v3.5.

Using WixNetfxExtension Properties

To use the WixNetfxExtension properties in an MSI, use the following steps:

- Add PropertyRef elements for items listed above that you want to use in your MSI.
- Add the -ext <path to WixNetfxExtension.dll> command line parameter when calling light.exe to include the WixNetfxExtension in the MSI linking process.

For example:

<PropertyRef Id="NETFRAMEWORK20" />

WixVSExtension

The <u>WixVSExtension</u> includes a set of custom actions to manage help collections. It also includes a set of properties and custom actions that can be used to detect the presence of various versions of Visual Studio and register add-ins, project templates and item templates for use in Visual Studio.

Properties

Here is a complete list of properties for the **Visual Studio .NET 2003** product family:

Property name	Meaning
	Full path to devenv.exe for Visual Studio .NET 2003 if it is installed on the system.
	Indicates whether or not the J# redistributable package 1.1 is installed on the system.

Here is a complete list of properties for the ${\bf Visual\ Studio\ 2005}$ product family:

Property name	Meaning
VS2005DEVENV	Full path to devenv.exe f Visual Studic 2005 if it is installed on t system.
VS2005_ITEMTEMPLATES_DIR	Full path to t Visual Studic 2005 item templates directory.
VS2005_PROJECTTEMPLATES_DIR	Full path to t

	Visual Studic 2005 project templates directory.
VS2005_SCHEMAS_DIR	Full path to t Visual Studic 2005 XML schemas directory.
VS2005PROJECTAGGREGATOR2	Indicates whether or n the Visual Studio 2005 project aggregator formanaged co add-ins is installed on t system.
VS2005_ROOT_FOLDER	Full path to t Visual Studic 2005 root installation directory.
VB2005EXPRESS_IDE	Full path to vbexpress.e: if Visual Bas 2005 Expres Edition is installed on t system.
VS2005_IDE_VB_PROJECTSYSTEM_INSTALLED	Indicates

	Studio 2005 Standard Edition or higher is installed and the Visual Ba project syste is installed fo
VC2005EXPRESS_IDE	Full path to vcexpress.ex Visual C++ 2005 Expres Edition is installed on t system.
VS2005_IDE_VC_PROJECTSYSTEM_INSTALLED	Indicates whether Visu Studio 2005 Standard Edition or higher is installed and the Visual C- project syste is installed fo
VCSHARP2005EXPRESS_IDE	Full path to vcsexpress.6 if Visual C# 2005 Expres Edition is installed on t system.

VS2005_IDE_VCSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Visuostudio 2005 Standard Edition or higher is installed and the Visual Ciproject syste is installed fo
VJSHARP2005EXPRESS_IDE	Full path to vjsexpress.e if Visual J# 2005 Expres Edition is installed on t system.
VS2005_IDE_VJSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Visu Studio 2005 Standard Edition or higher is installed and the Visual J# project syste is installed fc
VWD2005EXPRESS_IDE	Full path to vwdexpress. if Visual Wek Developer 20 Express Edit is installed o the system.

VS2005_IDE_VWD_PROJECTSYSTEM_INSTALLED	Indicates whether Visu Studio 2005 Standard Edition or higher is installed and the Visual W Developer project syste is installed fo
VS2005_IDE_VSTS_TESTSYSTEM_INSTALLED	Indicates whether or n the Visual Studio Team Test project system is installed on t system.
VSEXTENSIONS_FOR_NETFX30_INSTALLED	Indicates whether or n the Visual Studio 2008 Developmen Tools for the .NET Framework { add-in for Vis Studio 2005 installed on t system.
VS2005_WAP_PROJECT_INSTALLED	Indicates whether or n the Web Application

	Project template for Visual Studic 2005 is insta on the syster This project template is available as standalone a in and as a p of visual Stu- 2005 SP1.
VS2005_SP_LEVEL	Indicates the service pack level for Visu Studio 2005 Standard Edition and higher.
VSTF2005_SP_LEVEL	Indicates the service pack level for Visu Studio 2005 Team Foundation.
VB2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visu Basic 2005 Express Edition.
VC2005EXPRESS_SP_LEVEL	Indicates the service pack

	level for Visu C++ 2005 Express Edition.
VCSHARP2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visu C# 2005 Express Edition.
VJSHARP2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visu J# 2005 Express Edition.
VWD2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visu Web Develo 2005 Expres Edition.
DEXPLORE_2005_INSTALLED	Indicates whether or n the Documer Explorer 20C runtime components package is installed on t system.
JSHARP_REDIST_20_INSTALLED	Indicates

	whether or n the J# redistributab package 2.0 installed on t system.
JSHARP_REDIST_20SE_INSTALLED	Indicates whether or n the J# redistributab package 2.0 second editic is installed o the system.

Here is a complete list of properties for the **Visual Studio 2008** product family:

Property name	Meaning
VS90DEVENV	Full path to devenv.exe for Visual Studio 2008 if it is installed on the system.
VS90_ITEMTEMPLATES_DIR	Full path to the Visual Studio 2008 item templates directory.
VS90_PROJECTTEMPLATES_DIR	Full path to the Visual Studio 2008 project

	templates directory.
VS90_SCHEMAS_DIR	Full path to the Visual Studio 2008 XML schemas directory.
VS90_ROOT_FOLDER	Full path to the Visual Studio 2008 root installation directory.
VB90EXPRESS_IDE	Full path to vbexpress.exe if Visual Basic 2008 Express Edition is installed on the system.
VS90_IDE_VB_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2008 Standard Edition or higher is installed and the Visual Basi project system is installed for i
VC90EXPRESS_IDE	Full path to vcexpress.exe Visual C++

	2008 Express Edition is installed on the system.
VS90_IDE_VC_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2008 Standard Edition or higher is installed and the Visual C++ project system is installed for i
VCSHARP90EXPRESS_IDE	Full path to vcsexpress.exe if Visual C# 2008 Express Edition is installed on the system.
VS90_IDE_VCSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2008 Standard Edition or higher is installed and the Visual C# project system is installed for i
VWD90EXPRESS_IDE	Full path to vwdexpress.ex

	if Visual Web Developer 200 Express Edition is installed on the system.
VS90_IDE_VWD_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2008 Standard Edition or higher is installed and the Visual Web Developer project system is installed for i
VS90_IDE_VSTS_TESTSYSTEM_INSTALLED	Indicates whether or not the Visual Studio Team Test project system is installed on the system.
VS90_BOOTSTRAPPER_PACKAGE_FOLDER	The location of the Visual Studio 2008 bootstrapper package folder
VS90_SP1	Indicates whether or not service pack 1 for Visual

	Studio 2008 Standard Edition and higher is installed.
VS90_SP_LEVEL	Indicates the service pack level for Visual Studio 2008 Standard Edition and higher.
VSTF90_SP_LEVEL	Indicates the service pack level for Visual Studio 2008 Team Foundation.
VB90EXPRESS_SP_LEVEL	Indicates the service pack level for Visual Basic 2008 Express Edition.
VB90EXPRESS_SP1	Indicates whether or not service pack 1 for Visual Basic 2008 Express Edition is installed.
VC90EXPRESS_SP_LEVEL	Indicates the

	service pack level for Visual C++ 2008 Express Edition.
VC90EXPRESS_SP1	Indicates whether or not service pack 1 for Visual C++ 2008 Express Edition is installed.
VCSHARP90EXPRESS_SP_LEVEL	Indicates the service pack level for Visual C# 2008 Express Edition.
VCSHARP90EXPRESS_SP1	Indicates whether or not service pack 1 for Visual C# 2008 Express Edition is installed.
VWD90EXPRESS_SP_LEVEL	Indicates the service pack level for Visual Web Develope 2008 Express Edition.
VWD90EXPRESS_SP1	Indicates

	whether or not service pack 1 for Visual Web Developer 200 Express Edition is installed.
DEXPLORE_2008_INSTALLED	Indicates whether or not the Document Explorer 2008 runtime components package is installed on the system.

Here is a complete list of properties for the **Visual Studio 2010** product family:

Property name	Meaning
VS2010DEVENV	Full path to devenv.exe Visual Stud 2010 if it is installed on system. Thi property is available starting with WiX v3.5.
VS2010_ITEMTEMPLATES_DIR	Full path to Visual Stud 2010 item templates

	directory. To property is available starting with WiX v3.5.
	Full path to Visual Stud 2010 projec templates directory. To property is available starting with WiX v3.5.
VS2010_SCHEMAS_DIR	Full path to Visual Stud 2010 XML schemas directory. T property is available starting with WiX v3.5.
	Full path to Visual Stud 2010 root installation directory. T property is available starting with WiX v3.5.
VB2010EXPRESS_IDE	Full path to

vbexpress.e if Visual Ba 2010 Expre Edition is installed on system. Thi property is available starting with WiX v3.5.
Indicates whether Vis Studio 201(Standard Edition or higher is installed an the Visual E project syst is installed This proper available starting with WiX v3.5.
Full path to vcexpress.e Visual C++ 2010 Expre Edition is installed on system. Thi property is available starting with WiX v3.5.

VS2010_IDE_VC_PROJECTSYSTEM_INSTALLED	Indicates whether Vis Studio 201(Standard Edition or higher is installed an the Visual (project syst is installed This proper available starting with WiX v3.5.
VCSHARP2010EXPRESS_IDE	Full path to vcsexpress if Visual C# 2010 Expre Edition is installed on system. Thi property is available starting with WiX v3.5.
VS2010_IDE_VCSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Vis Studio 201(Standard Edition or higher is installed an the Visual (project syst is installed This proper

	available starting with WiX v3.5.
VWD2010EXPRESS_IDE	Full path to vwdexpress if Visual We Developer 2 Express Ed is installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_VWD_PROJECTSYSTEM_INSTALLED	Indicates whether Vis Studio 201(Standard Edition or higher is installed an the Visual V Developer project syst is installed This proper available starting with WiX v3.5.
VPD2010EXPRESS_IDE	Full path to vpdexpress if Visual Stu 2010 Expre for Window Phone is

	installed on system. Thi property is available starting with WiX v3.5.
VS2010_IDE_VSTS_TESTSYSTEM_INSTALLED	Indicates whether or the Visual Studio 2010 Team Test project syst is installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_DB_PROJECTSYSTEM_INSTALLED	Indicates whether or the Visual Studio 2010 Database project syst is installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_VSD_PROJECTSYSTEM_INSTALLED	Indicates whether or the Visual Studio 201(Deploymen

	project syst (setup projetis installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_WIX_PROJECTSYSTEM_INSTALLED	Indicates whether or the Visual Studio 201(Windows Installer XN project syst is installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_MODELING_PROJECTSYSTEM_INSTALLED	Indicates whether or the Visual Studio 201(Modeling project syst is installed the system. This proper available starting with WiX v3.5.
VS2010_IDE_FSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether or

the Visual Studio 201(project syst is installed the system. This proper available starting with WiX v3.5.
The location the Visual Studio 2010 bootstrappe package fol This proper available starting with WiX v3.5.

Custom Actions

Here is a complete list of custom actions:

Custom action name	Meaning
VS2003Setup	Runs devenv.exe /setup if a Visual Studio .NET 2003 edition is found on the system.
VS2005Setup	Runs devenv.exe /setup if Visual Studio 2005 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS2005DEVENV property.
VS2005InstallVSTemplates	Runs devenv.exe /InstallVSTemplates if Visual Studio 2005 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS2005DEVENV property.
VB2005Setup	Runs vbexpress.exe /setup if Visual Basic 2005 Express Edition is found on the system. Including this custom action automatically adds the VB2005EXPRESS_IDE property.
VB2005InstallVSTemplates	Runs vbexpress.exe /InstallVSTemplates if Visual Basic 2005 Express Edition is found on the system. Including this custom

	action automatically adds the VB2005EXPRESS_IDE property.
VC2005Setup	Runs vcexpress.exe /setup if Visual C++ 2005 Express Edition is found on the system. Including this custom action automatically adds the VC2005EXPRESS_IDE property.
VC2005InstallVSTemplates	Runs vcexpress.exe /InstallVSTemplates if Visual C++ 2005 Express Edition is found on the system. Including this custom action automatically adds the VC2005EXPRESS_IDE property.
VCSHARP2005Setup	Runs vcsexpress.exe /setup if Visual C# 2005 Express Edition is found on the system. Including this custom action automatically adds the VCSHARP2005EXPRESS_IDE property.
VCSHARP2005InstallVSTemplates	Runs vcsexpress.exe /InstallVSTemplates if Visual C# 2005 Express Edition is found on the system. Including this custom action automatically adds the VCSHARP2005EXPRESS_IDE property.
VJSHARP2005Setup	Runs vjsexpress.exe /setup if Visual J# 2005 Express Edition is found on the system. Including this custom action automatically adds the VJSHARP2005EXPRESS_IDE property.

<u>Į</u>	<u>J</u>
VJSHARP2005InstallVSTemplates	Runs vjsexpress.exe /InstallVSTemplates if Visual J# 2005 Express Edition is found on the system. Including this custom action automatically adds the VJSHARP2005EXPRESS_IDE property.
VWD2005Setup	Runs vwdexpress.exe /setup if Visual Web Developer 2005 Express Edition is found on the system. Including this custom action automatically adds the VWD2005EXPRESS_IDE property.
VWD2005InstallVSTemplates	Runs vwdexpress.exe /InstallVSTemplates if Visual Web Developer 2005 Express Edition is found on the system. Including this custom action automatically adds the VWD2005EXPRESS_IDE property.
VS90Setup	Runs devenv.exe /setup if Visual Studio 2008 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS90DEVENV property.
VS90InstallVSTemplates	Runs devenv.exe /InstallVSTemplates if Visual Studio 2008 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS90DEVENV property.

VB90Setup	Runs vbexpress.exe /setup if Visual Basic 2008 Express Edition is found on the system. Including this custom action automatically adds the VB90EXPRESS_IDE property.
VB90InstallVSTemplates	Runs vbexpress.exe /InstallVSTemplates if Visual Basic 2008 Express Edition is found on the system. Including this custom action automatically adds the VB90EXPRESS_IDE property.
VC90Setup	Runs vcexpress.exe /setup if Visual C++ 2008 Express Edition is found on the system. Including this custom action automatically adds the VC90EXPRESS_IDE property.
VC90InstallVSTemplates	Runs vcexpress.exe /InstallVSTemplates if Visual C++ 2008 Express Edition is found on the system. Including this custom action automatically adds the VC90EXPRESS_IDE property.
VCSHARP90Setup	Runs vcsexpress.exe /setup if Visual C# 2008 Express Edition is found on the system. Including this custom action automatically adds the VCSHARP90EXPRESS_IDE property.
VCSHARP90InstallVSTemplates	Runs vcsexpress.exe /InstallVSTemplates if Visual C# 2008 Express Edition is found on

	the system. Including this custom action automatically adds the VCSHARP90EXPRESS_IDE property.
VWD90Setup	Runs vwdexpress.exe /setup if Visual Web Developer 2008 Express Edition is found on the system. Including this custom action automatically adds the VWD90EXPRESS_IDE property.
VWD90InstallVSTemplates	Runs vwdexpress.exe /InstallVSTemplates if Visual Web Developer 2008 Express Edition is found on the system. Including this custom action automatically adds the VWD90EXPRESS_IDE property.
VS2010Setup	Runs devenv.exe /setup if Visual Studio 2010 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS2010DEVENV property. This custom action is available starting with WiX v3.5.
VS2010InstallVSTemplates	Runs devenv.exe /InstallVSTemplates if Visual Studio 2010 Standard Edition or higher is found on the system. Including this custom action automatically adds the VS2010DEVENV property. This custom action is available starting with WiX v3.5.

VB2010Setup	Runs vbexpress.exe /setup if Visual Basic 2010 Express Edition is found on the system. Including this custom action automatically adds the VB2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VB2010InstallVSTemplates	Runs vbexpress.exe /InstallVSTemplates if Visual Basic 2010 Express Edition is found on the system. Including this custom action automatically adds the VB2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VC2010Setup	Runs vcexpress.exe /setup if Visual C++ 2010 Express Edition is found on the system. Including this custom action automatically adds the VC2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VC2010InstallVSTemplates	Runs vcexpress.exe /InstallVSTemplates if Visual C++ 2010 Express Edition is found on the system. Including this custom action automatically adds the VC2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VCSHARP2010Setup	Runs vcsexpress.exe /setup if Visual C# 2010 Express Edition is found on the system. Including this

	custom action automatically adds the VCSHARP2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VCSHARP2010InstallVSTemplates	Runs vcsexpress.exe /InstallVSTemplates if Visual C# 2010 Express Edition is found on the system. Including this custom action automatically adds the VCSHARP2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VWD2010Setup	Runs vwdexpress.exe /setup if Visual Web Developer 2010 Express Edition is found on the system. Including this custom action automatically adds the VWD2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VWD2010InstallVSTemplates	Runs vwdexpress.exe /InstallVSTemplates if Visual Web Developer 2010 Express Edition is found on the system. Including this custom action automatically adds the VWD2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VPD2010Setup	Runs vpdexpress.exe /setup if Visual Studio 2010 Express for Windows Phone is found on the system. Including this custom action automatically adds the

	VPD2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.
VPD2010InstallVSTemplates	Runs vpdexpress.exe /InstallVSTemplates if Visual Studio 2010 Express for Windows Phone is found on the system. Including this custom action automatically adds the VPD2010EXPRESS_IDE property. This custom action is available starting with WiX v3.5.

Using WixVSExtension Properties or Custom Actions

To use the WixVSExtension properties or custom actions in an MSI, use the following steps:

- Add PropertyRef or CustomActionRef elements for items listed above that you want to use in your MSI.
- Add the -ext <path to WixVSExtension.dll> command line parameter when calling light.exe to include the WixVSExtension in the MSI linking process.

For example:

```
<PropertyRef Id="VS2005_ROOT_FOLDER" />
<CustomActionRef Id="VS2005Setup" />
```

When you reference any of the above properties or custom actions, the WixVSExtension automatically schedules the custom actions and pulls in properties used in the custom action conditions and execution logic.

Patch Building

Patches are updates to a product or products. WiX supports two different ways of creating them:

- <u>Using Patch Creation Properties</u> which requires that you have the Windows Installer 3.0 or newer SDK installed for full support of included examples.
- <u>Using Purely WiX</u> which uses functionality provided in WiX and does not require additional tools.

There are also <u>restrictions</u> on how patches are built in order to avoid problems when installing them.

How Patches Work

Patches contain a collection of transforms - most often a pair of transforms for each target product. When a patch is applied, each installed target product is reinstalled individually with the corresponding patch transforms applied. These transforms contain the differences between that target product and the upgrade product that might contain new file versions and sizes, new registry keys, etc.

For more information about patching with Windows Installer, read <u>Patching and Upgrades</u>.

Restrictions for Patches

There are different restrictions for patches based on what type of patch is to be installed. There are three types of patches:

- **Small updates** do not change the ProductVersion property of a target product and typically represent a small subset of files to be updated.
- **Minor upgrades** do change the ProductVersion property of a target product and typically represent a larger subset of files to be updated. Minor upgrades might also be installed as upgrade MSIs.
- Major upgrades change both the ProductVersion and ProductCode and contain all files in a product. Shipping major upgrades as a patch is, however, not recommended and WiX does not support building major upgrade patches because of the problems they create.

For information about restrictions for each type of patch, read <u>Changing the Product Code</u>.

Uninstallable Patches

For a patch to be uninstallable, the MsiPatchMetadata table must exist in the patch package and must contain the AllowRemoval property set to 1. This can be authored into the <u>Patch Creation Properties</u> file using the <u>PatchMetadata</u>/@AllowRemoval attribute or into the <u>patch XML</u> file using the <u>Patch</u>/@AllowRemoval attribute.

Beside that, certain tables cannot be modified in the upgrade package from which a patch is built. Read <u>Uninstallable Patches</u> for the current list of tables. Pyro.exe will error if one of these tables would be modified when building a <u>patch XML</u> file.

The following table lists tables and corresponding elements or attributes in WiX.

Table	Element or Attribute
BindImage	File/@BindPath
Class	<u>Class</u>
Complus	Component/@ComPlusFlags
CreateFolder	<u>CreateFolder</u>
DuplicateFile	<u>CopyFile</u>
Environment	<u>Environment</u>
Extension	<u>Extension</u>
Font	<u>File</u> /@FontTitle

IniFile	<u>IniFile</u>
IsolatedComponent	<u>IsolatedComponent</u>
LockPermissions	<u>Permission</u>
MIME	MIME
MoveFile	<u>CopyFile</u>
ODBCAttribute	ODBCDriver/Property
ODBCDataSource	<u>ODBCDataSource</u>
ODBCDriver	<u>ODBCDriver</u>
ODBCSourceAttribute	ODBCDataSource/Property
ODBCTranslator	<u>ODBCTranslator</u>
Progld	<u>Progld</u>
PublishComponent	<u>Category</u>
RemovelniFile	<u>IniFile</u>
SelfReg	File/@SelfRegCost
ServiceControl	<u>ServiceControl</u>
ServiceInstall	<u>ServiceInstall</u>
TypeLib	TypeLib

Verb	<u>Verb</u>	
	II .	

Major upgrade patches are not uninstallable.

Using Patch Creation Properties

A patch contains the differences between one or more pairs of Windows Installer packages. The tool PatchWiz.dll in the Windows SDK compares pairs of packages and produces a patch using a file called a Patch Creation Properties (PCP) file.

It is recommended that you download the latest Windows SDK to get the newest tools for building patches.

Setting Up the Sample

A Patch Creation Properties (PCP) file instructs PatchWiz.dll to generate a patch from differences in one or more pairs of packages. A patch contains the differences between the target and upgrade packages, and will transform the target package to the upgrade package. Both the target and upgrade packages are created below.

Create a directory that will contain the sample

Create a directory from which you plan on running the sample. This will be the sample root.

md C:\sample

Create two subdirectories

Under the sample root create two subdirectories called "1.0" and "1.1".

md C:\sample\1.0 md C:\sample\1.1

Create a text file called Sample.txt for 1.0

Create a text file in the "1.0" directory called Sample.txt and put some text in it telling you that it is the 1.0 version of the file.

echo This is version $1.0 > C:\sample 1.0 \le ample txt$

Create a text file called Sample.txt for 1.1

Create a text file in the "1.1" directory called Sample.txt and put some text in it telling you that it is the 1.1 version of the file.

echo This is version $1.1 > C:\sample \ 1.1\Sample.txt$

Create your product authoring in the sample root folder

Create your product authoring in the sample root folder called Product.wxs with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="48C49ACE-90CF-4161-9C6E-9162115A54DD"</p>
    Name="WiX Patch Example Product"
    Language="1033"
    Version="1.0.0"
    Manufacturer="Dynamo Corporation"
    UpgradeCode="48C49ACE-90CF-4161-9C6E-9162115A54DD">
    <Package Description="Installs a file that will be patched."</p>
      Comments="This Product does not install any executables"
      InstallerVersion="200"
      Compressed="yes" />
    <Media Id="1" Cabinet="product.cab" EmbedCab="yes" />
    <FeatureRef Id="SampleProductFeature"/>
  </Product>
  <Fragment>
    <Feature Id="SampleProductFeature" Title="Sample Product Feature" Leve</p>
      <ComponentRef Id="SampleComponent" />
    </Feature>
  </Fragment>
  <Fragment>
    <DirectoryRef Id="SampleProductFolder">
      <Component Id="SampleComponent" Guid="{C28843DA-EF08-41CC-</p>
         <File Id="SampleFile" Name="Sample.txt" Source=".\$(var.Version)\5</pre>
      </Component>
    </DirectoryRef>
  </Fragment>
  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder" Name="PFiles">
```

Create your patch authoring in the sample root

Create your Patch Creation Properties (PCP) authoring in the sample root called Patch.wxs with the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <PatchCreation
    Id="224C316C-5894-4771-BABF-21A3AC1F75FF"
    CleanWorkingFolder="yes"
    OutputPath="patch.pcp"
    WholeFilesOnly="yes"
    >
    < Patch Information
      Description="Small Update Patch"
      Comments="Small Update Patch"
      ShortNames="no"
      Languages="1033"
      Compressed="yes"
      Manufacturer="Dynamo Corp"/>
    <PatchMetadata
      AllowRemoval="yes"
      Description="Small Update Patch"
      ManufacturerName="Dynamo Corp"
      TargetProductName="Sample"
      MoreInfoURL="http://www.dynamocorp.com/"
      Classification="Update"
      DisplayName="Sample Patch"/>
    <Family DiskId="5000"</pre>
```

Note that **SequenceStart** must be greater than the last sequence in the File table in the target package or the patch will not install.

Build the Target and Upgrade Packages

Open a command prompt and make sure the following WiX and Windows Installer SDK tools are in your PATH.

- Candle.exe
- Light.exe
- MsiMsp.exe
- PatchWiz.dll
- MSPatchC.dll
- MakeCab.exe

Build the target package

candle.exe -dVersion=1.0 product.wxs light.exe product.wixobj -out 1.0\product.msi

Perform an administrative installation of the target package

Msiexec.exe is used to perform an administrative installation but nothing is actually registered on your system. It is mainly file extraction.

msiexec.exe /a 1.0\product.msi /qb TARGETDIR=C:\sample\1.0\admin

Build the upgrade package

candle.exe -dVersion=1.1 product.wxs
light.exe product.wixobj -out 1.1\product.msi

Perform an administrative installation of the upgrade package

msiexec.exe /a 1.1\product.msi /qb TARGETDIR=C:\sample\1.1\admin

Build the Patch

The Patch.wxs file is compiled into a PCP file that is then processed by MsiMsp.exe to product the patch package.

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

Verify the Patch

To verify that the patch works, install the product and then the patch.

Install the 1.0 product

msiexec.exe /i 1.0\product.msi /l*vx install.log

Verify version 1.0

Go to "Program Files\Patch Sample Directory" and open Sample.txt. Verify that this is the 1.0 version. Close Sample.txt.

Install the patch

msiexec.exe /p patch\patch.msp /l*vx patch.log

Verify version 1.1

Go to "Program Files\Patch Sample Directory" and open Sample.txt. Verify that this is now the 1.1 version. Close Sample.txt.

Uninstall the patch

On Windows XP Service Pack 2 and Windows Server 2003, go to "Add/Remove Programs" in the Control Panel and make sure that Show Updates is checked. On Windows Vista and newer, go to "Programs" then "View installed updates" in the Control panel. Select "Sample Patch" from under "WiX Patch Example Product" and click the Uninstall button.

Go to "Program files\Patch Sample Directory" and open Sample.txt. Verify that this is again the 1.0 version. Close Sample.txt.

Uninstall the product

On Windows XP Service Pack 2 and Windows Server 2003, go to "Add/Remove Programs" in the Control Panel. On Windows Vista and

newer, go to "Programs" then "Uninstall a program" in the Control Panel. Select "WiX Patch Example Product" and click the Uninstall button.

Restrictions

Please review <u>restrictions</u> on how patches must be built to avoid problem during patch installation.

Using Purely WiX

A patch can be created purely in WiX using the tools named Torch.exe and Pyro.exe. Using these tools eliminates the need to perform administrative installs or even to bind the upgrade product which, for large products, can be exhausting.

Setting Up the Sample

A sample product is created which puts different resources into fragments. You put resources into separate fragments so that the resources in each fragment can be filtered out of a patch. You might filter some resources out of a patch if you want to limit the patch to update only parts of your product or products.

Create a directory that will contain the sample

Create a directory from which you plan to run the sample. This will be the sample root.

md C:\sample

Create two subdirectories

Under the sample root create two subdirectories called "1.0" and "1.1".

md C:\sample\1.0 md C:\sample\1.1

Create a text file called Sample.txt for 1.0

Create a text file in the "1.0" directory called Sample.txt and put some text in it telling you that it is the 1.0 version of the file.

echo This is version $1.0 > C:\sample 1.0 \le ample txt$

Create a text file called Sample.txt for 1.1

Create a text file in the "1.1" directory called Sample.txt and put some text in it telling you that it is the 1.1 version of the file.

echo This is version $1.1 > C:\sample \ 1.1\Sample.txt$

Create your product authoring in the sample root folder

Create your product authoring in the sample root folder called Product.wxs with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Product Id="48C49ACE-90CF-4161-9C6E-9162115A54DD"</p>
    Name="WiX Patch Example Product"
    Language="1033"
    Version="1.0.0"
    Manufacturer="Dynamo Corporation"
    UpgradeCode="48C49ACE-90CF-4161-9C6E-9162115A54DD">
    <Package Description="Installs a file that will be patched."</p>
      Comments="This Product does not install any executables"
      InstallerVersion="200"
      Compressed="yes" />
    <Media Id="1" Cabinet="product.cab" EmbedCab="yes" />
    <FeatureRef Id="SampleProductFeature"/>
  </Product>
  <Fragment>
    <Feature Id="SampleProductFeature" Title="Sample Product Feature" Leve</p>
      <ComponentRef Id="SampleComponent" />
    </Feature>
  </Fragment>
  <Fragment>
    <DirectoryRef Id="SampleProductFolder">
      <Component Id="SampleComponent" Guid="{C28843DA-EF08-41CC-</p>
         <File Id="SampleFile" Name="Sample.txt" Source=".\$(var.Version)\5</pre>
      </Component>
    </DirectoryRef>
  </Fragment>
  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder" Name="PFiles">
```

Create your patch authoring in the sample root

Create your patch authoring in the sample root called Patch.wxs with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Patch
    AllowRemoval="yes"
    Manufacturer="Dynamo Corp"
    MoreInfoURL="http://www.dynamocorp.com/"
    DisplayName="Sample Patch"
    Description="Small Update Patch"
    Classification="Update"
    >
    <Media Id="5000" Cabinet="RTM.cab">
      <PatchBaseline Id="RTM"/>
    </Media>
    <PatchFamilyRef Id="SamplePatchFamily"/>
  </Patch>
  <Fragment>
    <PatchFamily Id='SamplePatchFamily' Version='1.0.0.0' Supersede='yes'>
      <ComponentRef Id="SampleComponent"/>
    </PatchFamily>
  </Fragment>
</Wix>
```

Building the Patch Sample

Open a command prompt and make sure that the following WiX tools are in your PATH.

- Candle.exe
- Light.exe
- Torch.exe
- Pyro.exe

Your WiX toolset version should be at least 3.0.3001.0

Build the target layout

While only the .wixout is needed, the target product layout is created to test installing the patch. The product must also be installed before or along with the patch.

```
cd C:\sample candle.exe -dVersion=1.0 product.wxs light.exe product.wixobj -out 1.0\product.msi
```

Build the upgrade layout

```
candle.exe -dVersion=1.1 product.wxs light.exe product.wixobj -out 1.1\product.msi
```

Create the transform between your products

 $torch. exe -p -xi \ 1.0 \\ \ product. \\ wixpdb \ 1.1 \\ \ product. \\ wixpdb -out \ patch \\ \ diff. \\ wixmst$

Build the patch

The patch.wxs file is compiled and linked like a product, but then it is processed along with any number of transforms that you want the patch to contain. That produces an MSP file that targets any of the products

from which transforms were created after filtering.

candle.exe patch.wxs
light.exe patch.wixobj -out patch\patch.wixmsp
pyro.exe patch\patch.wixmsp -out patch\patch.msp -t RTM patch\diff.wixmst

Verify the Patch

To verify that the patch works, install the product and then the patch.

Install the 1.0 product

msiexec.exe /i 1.0\product.msi /l*vx install.log

Verify version 1.0

Go to "Program Files\Patch Sample Directory" and open Sample.txt. Verify that this is the 1.0 version. Close Sample.txt.

Install the patch

msiexec.exe /p patch\patch.msp /l*vx patch.log

Verify version 1.1

Go to "Program Files\Patch Sample Directory" and open Sample.txt. Verify that this is now the 1.1 version. Close Sample.txt.

Uninstall the patch

On Windows XP Service Pack 2 and Windows Server 2003, go to "Add/Remove Programs" in the Control Panel and make sure that Show Updates is checked. On Windows Vista and newer, go to "Programs" then "View installed updates" in the Control Panel. Select "Sample Patch" from under "WiX Patch Example Product" and click the Uninstall button.

Go to "Program files\Patch Sample Directory" and open Sample.txt. Verify that this is again the 1.0 version. Close Sample.txt.

Uninstall the product

On Windows XP Service Pack 2 and Windows Server 2003, go to "Add/Remove Programs" in the Control Panel. On Windows Vista and

newer, go to "Programs" then "Uninstall a program" in the Control Panel. Select "WiX Patch Example Product" and click the Uninstall button.

Restrictions

In addition to <u>restrictions</u> about what can be in a patch in order for it to install and uninstall correctly, the following restrictions ensure that your patch works correctly.

Patch families can only grow

Patch families are used to filter resources that should end up in a patch. Once the patch is created, these patch families dictate which patches are superseded. If a resource is removed from a patch family in a newer patch and that resource is contained in an older patch with the same patch family, then when the older patch is superseded, that resource will be regressed back to its previous state before the older patch was installed.

Note that in order for one patch to supersede any other patches, all patch families must be superseded. A single patch family is referenced in the example above for simplicity.

Certain elements cannot be added to uninstallable patches

There are certain elements referenced in <u>restrictions</u> that cannot be added or modified if the patch is to be uninstallable. If a Patch/@AllowRemoval is set to "yes" and any of these elements are added or modified, Pyro.exe will return an error. These elements compile into tables that Windows Installer restricts in patches, so WiX informs you and prevents you from creating a patch that is not uninstallable when you want it to be uninstallable.

WiX Schema References

This section contains schema reference information for WiX and extensions.

- Wix schema
- Wixloc schema
- Difxapp schema for WixDifxAppExtension
- Firewall schema for WixFirewallExtension
- Gaming schema for WixGamingExtension
- <u>lis schema for WixIIsExtension</u>
- IsolatedApp schema for WixIsolatedAppExtension
- Lux schema for WixLuxExtension
- Netfx schema for WixNetFxExtension
- OfficeAddin schema for WixOfficeExtension
- Ps schema for WixPSExtension
- Sql schema for WixSqlExtension
- Util schema for WixUtilExtension
- Vs schema for WixVSExtension

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/2006/wi

Document Should Look Like

```
<!xml version="1.0"?>
<!nclude xmlns="http://schemas.microsoft.com/wix/2006/wi">
.
.
.
.
./Include>
<!xml version="1.0"?>
<!wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.</Wix>
```

AdminExecuteSequence Element

Description

None

Windows Installer references

AdminExecuteSequence Table

Parents

Fragment, Module, Product

Inner Text

None

Children

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

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

- <u>LaunchConditions</u> (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

AdminUISequence Element

Description

None

Windows Installer references

<u>AdminUISequence Table</u>

Parents

Fragment, Module, Product, UI

Inner Text

None

Children

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

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

- <u>InstallValidate</u> (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.
- <u>LaunchConditions</u> (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

AdvertiseExecuteSequence Element

Description

None

Windows Installer references

AdvtExecuteSequence Table

Parents

Fragment, Module, Product

Inner Text

None

Children

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

- <u>CostFinalize</u> (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- <u>CostInitialize</u> (min: 0, max: unbounded): Initiates the internal installation costing process.
- <u>CreateShortcuts</u> (min: 0, max: unbounded): Manages the creation of shortcuts.
- <u>Custom</u> (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).
- <u>InstallFinalize</u> (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- <u>InstallInitialize</u> (min: 0, max: unbounded): Marks the beginning of a sequence of actions that change the system.
- <u>InstallValidate</u> (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.
- <u>MsiPublishAssemblies</u> (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.

- <u>PublishFeatures</u> (min: 0, max: unbounded): Writes each feature's state into the system registry.
- <u>PublishProduct</u> (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.
- RegisterProgldInfo (min: 0, max: unbounded): Manages the registration of OLE Progld information with the system.

Attributes

None

See Also

Wix Schema

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

AllocateRegistrySpace Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Appld 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

Appld Table, Registry Table

Parents

Component, File, Fragment, Module, Product, TypeLib

Inner Text

None

Children

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

• Class (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	Guid	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	

		Appld 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.
DIISurrogate	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 Appld element correspond to registry keys as follows (values that can be specified in authoring are in bold):

ld

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 Appld Description"

DIISurrogate

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

AppSearch Table, AppSearch Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Туре	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.	

Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema, ComponentSearch, FileSearch, IniFileSearch, RegistrySearch

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	Туре	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. <u>Control</u> (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	Туре	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. <u>Billboard</u> (min: 1, max: unbounded): Order of Billboard elements determines order of display

Attributes

Name	Туре	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, Module, Product, UI

Inner Text

None

Children

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

• Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Attributes

Name	Туре	Description	Required
Id	String	The Id cannot by 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.	
SuppressModularization	YesNoType	Use to suppress modularization of this Binary identifier in merge modules.	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema

BinaryRef Element

Description

Used only for PatchFamilies to include only a binary table entry in a patch.

Windows Installer references

None

Parents

PatchFamily

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The identifier of the Binary element to reference.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

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

BindImage Table, BindImage Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

Wix Schema

Bundle Element

Description

The root element for creating bundled packages.

Windows Installer references

None

Parents

<u>Wix</u>

Inner Text

None

Children

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

- Chain (min: 1, max: 1)
- Container (min: 0, max: unbounded)
- ContainerRef (min: 0, max: unbounded)
- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- <u>UX</u> (min: 1, max: 1)
- Variable (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - Chain
 - ComponentSearch
 - Container
 - ContainerRef
 - DirectorySearch
 - FileSearch
 - LayoutDirectory
 - LayoutDirectoryRef

- ProductSearch
- RegistrySearch
- <u>UX</u>
- <u>Variable</u>

Attributes None

See Also

Wix Schema

Category Element

Description

Qualified published component for parent Component

Windows Installer references

PublishComponent Table

Parents

Component

Inner Text

None

Children

Sequence (min: 0, max: unbounded)

1. AppData (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	Guid	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	Yes

in the Id attribute. A qualifier is used to distinguish multiple forms of the same Component, such as a Component that is implemented in multiple languages.

See Also

Wix Schema

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

CCPSearch Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	

Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema, RMCCPSearch, ComplianceCheck

Chain Element

Description

Contains the chain of packages to install.

Windows Installer references

None

Parents

Bundle

Inner Text

None

Children

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

- ExePackage (min: 0, max: unbounded)
- MsiPackage (min: 0, max: unbounded)
- MspPackage (min: 0, max: unbounded)
- MsuPackage (min: 0, max: unbounded)
- PackageGroupRef (min: 0, max: unbounded)

Attributes

None

See Also

Wix Schema

Class Element

Description

COM Class registration for parent Component.

Windows Installer references

Class Table, Progld Table, Registry Table, Appld Table

Parents

Appld, Component, File, 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).
- Progld (min: 0, max: unbounded): A Progld associated with Class must be a child element of the Class element

Attributes

Name	Туре	Description	Required
Id	Guid	The Class identifier (CLSID) of a COM server.	Yes
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 often problematic Windows Installer advertising behavior).
Appld	Guid	This attribute is only allowed when a Class is advertised. Using this attribute will reference an Application ID containing DCOM information for the associated application GUID. The value must correspond to an Appld/@Id of an Appld element nested under a Fragment, Module, or Product element. To associate an Appld with a non-advertised class, nest the class within a parent Appld element.
Argument	String	This column is optional only when the Context column is set to "LocalServer" or "LocalServer32" server context. The text is registered as the argument against the OLE server and is used by OLE for invoking the server. 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 class 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 class.

Context

List

The server context(s) for this COM server. This attribute is optional for VB6 libraries that are marked "PublicNotCreateable". Class elements marked Advertised must specify at least one server context. It is most common for there to be a single value for the Context attribute. This attribute's value should be a spacedelimited list containg one or more of the

following: LocalServer

A 16-bit local server application.

LocalServer32
A 32-bit local server application.

InprocServer
A 16-bit inprocess server
DLL.

InprocServer32
A 32-bit inprocess server
DLL.

Control

YesNoType
Set this attribute's value to 'yes' to identify an object as an ActiveX Control.
The default value is

'no'.

Description String Localized description

associated with the Class ID and Program

ID.

ForeignServer String May only be specified

if the value of the Advertise attribute is "no" and Server has not been specified. In addition, it may only be used when the Class element is directly under the Component element.

The value can be that of an registry type (REG_SZ). This attribute should be used to specify foreign servers, such as mscoree.dll if needed.

Handler

String

The default inproc handler. May be optionally provided only for Context = LocalServer or 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 16-bit as well as 32-bit InprocHandlers. 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 if the value of the Advertise attribute is "no".
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 installer registers the file name only instead of the complete path. This enables the server in the current directory to take precedence and 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" and the ForeignServer attribute is not specified. File Id of the COM server file. If this element is nested under a File element, this value defaults to the value of the parent File/@Id.
ShortPath Yes!	oType Specifies whether or not to use the short path for the COM server. This can only apply when Advertise is set to 'no'. The default is 'no' meaning that it will use the long file name for the COM server.
ThreadingModel Enu	the CLSID. This attribute's value must be one of the following: apartment
	free
	both
	neutral
	single
	rental
Version Strir	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]"

Id/Context/ForeignServer

In General

[HKCR\CLSID\{Id}\Context1]

@="ForeignServer"

[HKCR\CLSID\{Id}\Context2]

@="ForeignServer"

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-

456789ABCDEF}\LocalServer]

@="mscoree.dll"

[HKCR\CLSID\{01234567-89AB-CDEF-0123-

456789ABCDEF}\LocalServer32]

@="mscoree.dll"

Appld

In General

[HKCR\CLSID\{**Id**}] Appld="{**Appld**}"

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
        456789ABCDEF}]
       Appld="{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 TypeLibI@Id)

In General

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

Specific Example

[HKCR\CLSID\{01234567-89AB-CDEF-0123-456789ABCDEF}\TypeLib] @="{1111111-89AB-1111-0123-11111111111}"

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, Appld

Column Element

Description

Column definition for a Custom Table

Windows Installer references

None

Parents

CustomTable

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Identifier for the column.	Yes
Category	Enumeration	Category of this column. This attribute must be specified with a value of 'Binary' if the Type attribute's value is 'binary'. This attribute's value must be one of the following: <i>Text</i>	
		UpperCase	
		LowerCase	
		Integer	
		DoubleInteger	
		TimeDate	

Identifier

Property

Filename

WildCardFilename

Path

Paths

AnyPath

DefaultDir

RegPath

Formatted

FormattedSddl

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	<u>YesNoType</u>	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	Enumeration	How this column should be modularized, if at all. This attribute's value must be one of the following: None Column should not be modularized. This is the default value.
		Column Column should be modularized.
		Condition Column is a condition and should be modularized.
		Icon When the column is an primary or foreign key to the Icon table it should be modularized special.
		Property Any Properties in the column should be modularized.
		SemicolonDelimited

Semi-colon list of keys, all of which need to be modularized.

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 must be one of the following: binary Column contains a path to a file that will be inserted into the column as a binary object. If this value is set, the Category attribute must also be set with a value of 'Binary' to pass ICE validation. int Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-localizable string value. Width Integer Width of this column.				
Set String Semicolon delimited list of permissible values. Type Enumeration The type of this column. This attribute's value must be one of the following: binary Column contains a path to a file that will be inserted into the column as a binary object. If this value is set, the Category attribute must also be set with a value of 'Binary' to pass ICE validation. int Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-localizable string value.	Nullable	<u>YesNoType</u>		
Type Enumeration The type of this column. This attribute's value must be one of the following: binary Column contains a path to a file that will be inserted into the column as a binary object. If this value is set, the Category attribute must also be set with a value of 'Binary' to pass ICE validation. int Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-localizable string value.	PrimaryKey	<u>YesNoType</u>		
This attribute's value must be one of the following: binary Column contains a path to a file that will be inserted into the column as a binary object. If this value is set, the Category attribute must also be set with a value of 'Binary' to pass ICE validation. int Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-localizable string value.	Set	String		
Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-localizable string value.	Туре	Enumeration	This attribute's value must be one of the following: binary Column contains a path to a file that will be inserted into the column as a binary object. If this value is set, the Category attribute must also be set with a value of 'Binary' to pass ICE	Yes
Width Integer Width of this column.			Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set). string Column contains a non-	
	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	Туре	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, Product

Inner Text

None

Children

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

- Sequence (min: 1, max: 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. <u>DirectorySearch</u> (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

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)

- <u>DirectorySearch</u> (min: 1, max: 1)
- <u>DirectorySearchRef</u> (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

ComponentGroup, Directory, DirectoryRef, Feature, FeatureGroup, FeatureRef, Fragment, Module, Product

Inner Text

None

Children

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

- Appld (min: 0, max: unbounded)
- Category (min: 0, max: unbounded)
- Class (min: 0, max: unbounded)
- Condition (min: 0, max: unbounded)
- CopyFile (min: 0, max: unbounded)
- <u>CreateFolder</u> (min: 0, max: unbounded)
- Environment (min: 0, max: unbounded)
- Extension (min: 0, max: unbounded)
- File (min: 0, max: unbounded)
- IniFile (min: 0, max: unbounded)
- Interface (min: 0, max: unbounded)
- <u>IsolateComponent</u> (min: 0, max: unbounded)
- ODBCDataSource (min: 0, max: unbounded)
- ODBCDriver (min: 0, max: unbounded)
- ODBCTranslator (min: 0, max: unbounded)
- Progld (min: 0, max: unbounded)
- Registry (min: 0, max: unbounded)

- RegistryKey (min: 0, max: unbounded)
- RegistryValue (min: 0, max: unbounded)
- RemoveFile (min: 0, max: unbounded)
- RemoveFolder (min: 0, max: unbounded)
- RemoveRegistryKey (min: 0, max: unbounded)
- RemoveRegistryValue (min: 0, max: unbounded)
- ReserveCost (min: 0, max: unbounded)
- <u>ServiceConfig</u> (min: 0, max: unbounded)
- <u>ServiceConfigFailureActions</u> (min: 0, max: unbounded)
- <u>ServiceControl</u> (min: 0, max: unbounded)
- <u>ServiceInstall</u> (min: 0, max: unbounded)
- Shortcut (min: 0, max: unbounded)
- SymbolPath (min: 0, max: unbounded)
- TypeLib (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - Certificate
 - ComPlusApplication
 - <u>ComPlusApplicationRole</u>
 - ComPlusAssembly
 - ComPlusGroupInApplicationRole
 - ComPlusGroupInPartitionRole
 - ComPlusPartition
 - ComPlusPartitionRole
 - ComPlusPartitionUser
 - <u>ComPlusRoleForComponent</u>
 - ComPlusRoleForInterface
 - ComPlusRoleForMethod
 - ComPlusSubscription
 - ComPlusUserInApplicationRole
 - <u>ComPlusUserInPartitionRole</u>
 - Driver

- EventSource
- FileShare
- FirewallException
- InternetShortcut
- MessageQueue
- MessageQueuePermission
- PerformanceCategory
- ServiceConfig
- SqlDatabase
- SqlScript
- SqlString
- <u>User</u>
- WebAppPool
- WebDir
- WebFilter
- WebProperty
- WebServiceExtension
- WebSite
- WebVirtualDir
- XmlConfig
- XmlFile

Name	Туре	Description
ComPlusFlags	Integer	Set this attribute to create a ComPlus entr. The value should be the export flags used during the generation of the life. For more information see the COM+documentation in the Platform SDK.
Directory	String	Sets the Directory of the

		Component. If this element is nested und a Directory element, the value defaults to the value of the parent Directory/@ld.
DisableRegistryReflection	YesNoType	Set this attribute to 'ye in order to disable registry reflection on a existing and new regis keys affected by this component. When set 'yes', the Windows Installer calls the RegDisableReflection on each key being accessed by the component. This bit is available with Window Installer version 4.0 ar is ignored on 32-bit systems.
DiskId	Integer	This attribute provides default DiskId attribute for all child File elemei Specifying the DiskId a Component element will override any DiskId attributes set by paren Directory or DirectoryF elements. See the File element's DiskId attrib for more information about the purpose of the DiskId.
Feature	String	Identifies a feature to which this component

belongs, as a shorthar for a child Component element of the Feature element. The value of this attribute should correspond to the Id attribute of a Feature element authored elsewhere. Note that a single component can belong to multiple features but this attribuallows you to specify ca single feature.

Guid

ComponentGuid

This value should be a guid that uniquely identifies this component's contents. language, platform, an version. If omitted, the default value is '*' which indicates that the linke should generate a stal guid. Generatable guid are supported only for components with a sin file as the component's keypath or no files and registry value as the keypath. It's also possible to set the value to an empty string to specify an unmanaged component. Unmanag components are a security vulnerability because the compone cannot be removed or

		repaired by Windows Installer (it is essential an unpatchable, permanent component Therefore, a guid shou always be specified for any component which contains resources the may need to be patched in the future.
Id	String	Component identifier; is the primary key for identifying component omitted, the compiler defaults the identifier t the identifier of the resource that is the explicit keypath of the component (for example a child File element will KeyPath attribute with value 'yes'.
KeyPath	YesNoType	If this attribute's value set to 'yes', then the Directory of this Component is used as the KeyPath. To set a Registry key or File as the KeyPath of a component, set the KeyPath attribute to 'yon one of those child elements.
Location	Enumeration	Optional value that specifies the location t the component can be run from. This attribute

value must be one of t following: local

Prevents the component from running from the source or the network (this is th default behavior if this attribute is no set).

source

Enforces that the component can or be run from the source (it cannot I run from the user' computer).

either

Allows the component to run from source or locally.

NeverOverwrite

<u>YesNoType</u>

If this attribute is set to 'yes', the installer does not install or reinstall the component if a key partile or a key path registentry for the component already exists. The application does regist itself as a client of the component. Use this flonly for components the are being registered by the Registry table. Do use this flag for

		components registered by the Appld, Class, Extension, Progld, MII and Verb tables.
Permanent	YesNoType	If this attribute is set to 'yes', the installer does not remove the component during an uninstall. The installer registers an extra syst client for the compone in the Windows Installer registry settings (which basically just means that least one product is always referencing this component). Note that this option differs from the behavior of not setting a guid because although the compone is permanent, it is still patchable (because Windows Installer still tracks it), it's just not uninstallable.
Shared	YesNoType	If this attribute's value set to 'yes', enables advanced patching semantics for Components that are shared across multiple Products. Specifically, the Windows Installer cache the shared files improve patch uninsta This functionality is available in Windows

		Installer 4.5 and later.
SharedDllRefCount	YesNoType	If this attribute's value set to 'yes', the installe increments the referer count in the shared DL registry of the component's key file. I this bit is not set, the installer increments the reference count only if the reference count already exists.
Transitive	YesNoType	If this attribute is set to 'yes', the installer reevaluates the value the statement in the Condition upon a reinstall. If the value w previously False and h changed to True, the installer installs the component. If the valu was previously True at has changed to False, the installer removes t component even if the component has other products as clients.
UninstallWhenSuperseded	YesNoType	If this attribute is set to 'yes', the installer will uninstall the Component's files and registry keys when it is superseded by a patch This functionality is available in Windows Installer 4.5 and later.

Win64

<u>YesNoType</u>

Set this attribute to 'ye to mark this as a 64-bi component. This attrib facilitates the installation of packages that incluboth 32-bit and 64-bit components. If this bit not set, the componen registered as a 32-bit component. If this is a 64-bit component replacing a 32-bit component replacing a 32-bit component, set this bit and assign a new GUI in the Guid attribute.

Any Attribute (namespace='##other' processContents='lax') Extensit the WiX XML Schema. Schema extensions can register additional at point in the schema.

How Tos and Examples

How To: Add a file to your installer

See Also

Wix Schema, ComponentRef, Media

ComponentGroup Element

Description

Groups together multiple components to be used in other locations.

Windows Installer references

None

Parents

Fragment, Product

Inner Text

None

Children

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

- Component (min: 0, max: unbounded)
- ComponentGroupRef (min: 0, max: unbounded)
- ComponentRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Attributes

Name	Туре	Description	Required
Id	String	Identifier for the ComponentGroup.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, ComponentGroupRef

ComponentGroupRef Element

Description

Create a reference to a ComponentGroup in another Fragment.

Windows Installer references

None

Parents

ComponentGroup, Feature, FeatureGroup, FeatureRef, Module

Inner Text

None

Children

None

Name	Туре	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.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, ComponentGroup

ComponentRef Element

Description

Create a reference to a Feature element in another Fragment.

Windows Installer references

None

Parents

ComponentGroup, Feature, FeatureGroup, FeatureRef, Module, PatchFamily

Inner Text

None

Children

None

Name	Туре	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.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

How Tos and Examples

• How To: Add a file to your installer

See Also

Wix Schema, Component

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)

• <u>DirectorySearch</u> (min: 0, max: 1)

• <u>DirectorySearchRef</u> (min: 0, max: 1)

• FileSearch (min: 0, max: 1)

• FileSearchRef (min: 0, max: 1)

Name	Туре	Description	Required
Id	String		Yes
Guid	Guid	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 must be one of the following: directory	

The key path of the component is a directory.

file

The key path of the component is a file. This is the default value.

See Also

Wix Schema, IniFileSearch, RegistrySearch

Condition Element

Description

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

Windows Installer references

<u>Component Table, ControlCondition Table, Condition Table, LaunchCondition Table</u>

Parents

Component, Control, Feature, Fragment, PermissionEx, 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

Name	Туре	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 must be one of the following: default Set the Control as the default. Only used under Control elements.	

enable

Enable the Control. Only used under Control elements.

disable

Disable the Control. Only used under Control elements.

hide

Hide the Control. Only used under Control elements.

show

Display the Control. Only used under Control elements.

Level	Integer	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.

How Tos and Examples

- How To: Block installation based on OS version
- How To: Check the version number of a file during installation

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

Name	Туре	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 must be one of the following: <i>Text</i>	Yes

Key Integer 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.	
Туре	String	Specifies the type of the data being changed.	

See Also

Wix Schema

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	Туре	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

Container Element

Description

Representation of a file that contains one or more files.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

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

• Payload (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Name	String	The destination path and file name for this container.	Yes

See Also

Wix Schema

ContainerRef Element

Description

Create a reference to an existing Container element.

Windows Installer references

None

Parents

Bundle

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The identifier of Container element to reference.	Yes

See Also

Wix Schema, Container

Control Element

Description

Contains the controls that appear on each dialog.

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. <u>Text</u> (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. <u>ListBox</u> (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)

Name	Туре	Description	Re
Id	String	Combined with the Dialog Id to make up the primary key of the Control table.	Υє
Bitmap	<u>YesNoType</u>	This attribute is only valid for RadioButton and PushButton Controls.	
Cancel	YesNoType	Set this attribute to "yes" to cause this Control to be invoked by the escape key.	
CDROM	<u>YesNoType</u>	This attribute is only valid for Volume and Directory Controls.	
CheckBoxPropertyRef	String	This attribute is only valid for CheckBox controls. The value is the name of a Property that was already used as the Property for another CheckBox control. The Property attribute cannot be specified. The attribute exists to support multiple checkboxes on different dialogs being tied to the same property.	
CheckBoxValue	String	This attribute is only	

		valid for CheckBox Controls. When set, the linked Property will be set to this value when the check box is checked.
ComboList	YesNoType	This attribute is only valid for ComboBox Controls.
Default	YesNoType	Set this attribute to "yes" to cause this Control to be invoked by the return key.
Disabled	YesNoType	Set this attribute to "yes" to cause the Control to be disabled.
ElevationShield	YesNoType	This attribute is only valid for PushButton controls. Set this attribute to "yes" to add 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	This attribute is only valid for Volume and Directory Controls.	
FixedSize	YesNoType	This attribute is only valid for RadioButton, PushButton, and Icon Controls.	
Floppy	<u>YesNoType</u>	This attribute is only valid for Volume and Directory Controls.	
FormatSize	<u>YesNoType</u>	This attribute is only valid for Text Controls.	
HasBorder	<u>YesNoType</u>	This attribute is only valid for RadioButton Controls.	
Height	LocalizableInteger	Height of the rectangular	Y€

		boundary of the control. This must be a non-negative number.
Help	String	This attribute is reserved for future use. There is no need to use this until Windows Installer uses it for something.
Hidden	YesNoType	Set this attribute to "yes" to cause the Control to be hidden.
Icon	YesNoType	This attribute is only valid for RadioButton and PushButton Controls.
IconSize	Enumeration	This attribute is only valid for RadioButton, PushButton, and Icon Controls. This attribute's value must be one of the following: 16 32 48
Image	YesNoType	This attribute is only valid for RadioButton, PushButton, and Icon Controls.

Indirect	YesNoType	Specifies whether the value displayed or changed by this control is referenced indirectly. If this bit is set, the control displays or changes the value of the property that has the identifier listed in the Property column of the Control table.
Integer	YesNoType	Set this attribute to "yes" to cause the linked Property value for the Control to be treated as an integer. Otherwise, the Property will be treated as a string.
LeftScroll	YesNoType	Set this attribute to "yes" to cause the scroll bar to display on the left side of the Control.
Multiline	<u>YesNoType</u>	This attribute is only valid for Edit Controls.
NoPrefix	<u>YesNoType</u>	This attribute is only valid for Text Controls.
NoWrap	<u>YesNoType</u>	This attribute is only valid for Text Controls.
Password	<u>YesNoType</u>	This attribute is only

		valid for Edit Controls.
ProgressBlocks	<u>YesNoType</u>	This attribute is only valid for ProgressBar Controls.
Property	String	The name of a defined property to be linked to this control. This column is required for active controls.
PushLike	<u>YesNoType</u>	This attribute is only valid for RadioButton and Checkbox Controls.
RAMDisk	<u>YesNoType</u>	This attribute is only valid for Volume and Directory Controls.
Remote	<u>YesNoType</u>	This attribute is only valid for Volume and Directory Controls.
Removable	<u>YesNoType</u>	This attribute is only valid for Volume and Directory Controls.
RightAligned	<u>YesNoType</u>	Set this attribute to "yes" to cause the Control to be right aligned.
RightToLeft	<u>YesNoType</u>	Set this attribute to "yes" to cause the Control to display from right to left.
ShowRollbackCost	<u>YesNoType</u>	This attribute is 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	<u>YesNoType</u>	Set this attribute to "yes" to cause the Control to be sunken.
TabSkip	YesNoType	Set this attribute to "yes" to cause this Control to be skipped in the tab sequence.
Text	String	A localizable string used to set the initial text contained in a control. This attribute can contain a formatted string that is processed at install time to insert the values of properties using [PropertyName]

		syntax. Also supported are environment variables, file installation paths, and component installation directories; see Formatted for details.	
ToolTip	String	The string used for the Tooltip.	
Transparent	YesNoType	This attribute is only valid for Text Controls.	
Type	String	The type of the control. Could be one of the following: Billboard, Bitmap, CheckBox, ComboBox, DirectoryCombo, DirectoryList, Edit, GroupBox, Hyperlink, Icon, Line, ListBox, ListView, MaskedEdit, PathEdit, ProgressBar, PushButton, RadioButtonGroup, ScrollableText, SelectionTree, Text, VolumeCostList, VolumeSelectCombo	Ye
UserLanguage	<u>YesNoType</u>	This attribute is only	

		valid for Text Controls.	
Width	LocalizableInteger	Width of the rectangular boundary of the control. This must be a non-negative number.	Υє
X	LocalizableInteger	Horizontal coordinate of the upper-left corner of the rectangular boundary of the control. This must be a non-negative number.	Yε
Y	LocalizableInteger	Vertical coordinate of the upper-left corner of the rectangular boundary of the control. This must be a non-negative number.	Ύє

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

Inner Text

None

Children

None

Name	Туре	Description
Id	String	Primary key u identify this paentry.

Delete	YesNoType	This attribute be specified if element is neumber a File e or the FileId a is specified. Ir cases, if the a is not specifie default value i and the file is not moved. Sevalue to "yes" to move the fi deleting the sefile) instead of it.
DestinationDirectory	String	Set this value destination directory and existence of the target of the should be more copied to. This directory must the installer directory at creation times attribute cannot specified in conjunction we destination of the destination of the stination of the s
DestinationLongName	LongFileNameType	This attribute been depreca please use the DestinationNa attribute inste
DestinationName	<u>LongFileNameType</u>	In prior versio WiX toolset, the attribute spec

short file name set this value localizable na given to the o file after it is n copied. If this is not specifie the destination given the sam as the source short file name specified, the DestinationSh attribute may specified. If a name is speci DestinationLo attribute may specified. Also value is a long name, the DestinationSh attribute may omitted to allo to attempt to g a unique shor name. Howev name collides another file or wish to manua specify the sh name, then th DestinationSh attribute may specified.

DestinationProperty

String

Set this value property that value that re

		to the full path destination did The property have to exist installer databcereation time; be created at installation time custom action command line This attribute be specified in conjunction w Destination Did
DestinationShortName	ShortFileNameType	The short file the file in 8.3 the This attribute only be set if the conflict betwe generated shound ames or you manually specified short file name
FileId	String	This attribute be specified if element is new under a File e Set this attribute value to the ic of a file from a different comparent component com
SourceDirectory	String	This attribute be specified if element is near

under a File e or the FileId a is specified. S value to the so directory from copy or move existing file or target machin Directory mus the installer do at creation timattribute cann specified in conjunction w SourceProper

SourceName

WildCardLongFileNameType This attribute

be specified if element is ne: under a File e or the FileId a is specified. S value to the localizable na the file(s) to b or moved. All files that matc wild card will I removed from specified direct The value is a filename that contain the wi characters "?" single charact for zero or mc occurrences c character. If th attribute is no

specified (and element is not under a File e or specify a F attribute) then SourceProper attribute shou to the name o property that \ resolve to the of the source filename. If the of this attribut contains a "*" and the DestinationNa attribute is sp all moved or c files retain the names from the sources.

SourceProperty

String

This attribute be specified if element is ne: under a File e or the FileId a is specified. S value to a pro that will have that resolves t path of the so directory (or fu including file r SourceName specified). The property does have to exist i installer datak

creation time; be created at installation tin custom action command line This attribute be specified in conjunction w SourceDirector

See Also

Wix Schema, RemoveFile

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

CostFinalize Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallUISequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the	

position of this action in a sequence.

Suppress <u>YesNoType</u> If yes, this action will not occur.

See Also

Wix Schema, CostInitialize, FileCost

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

CostInitialize Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallUISequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Inner Text

None

Children

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

- Permission (min: 0, max: unbounded): ACL permission
- <u>PermissionEx</u> (min: 0, max: unbounded): Can also configure the ACLs for this folder.
- Shortcut (min: 0, max: unbounded): Non-advertised shortcut to this folder, Shortcut Target is preset to the folder
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - PermissionEx

Attributes

Name	Туре	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

CreateFolders Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

CreateShortcuts Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallUISequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	Description	Required
Action	String	The action to which the Custom element applies.	Yes
After	String	The name of the standard or custom action after which this action should be performed. Mutually exclusive with Before, OnExit, and Sequence attributes	
Before	String	The name of the standard or custom action before which this action should be performed. Mutually exclusive with OnExit, After, and Sequence attributes	

OnExit	<u>ExitType</u>	Mutually exclusive with Before, After, and Sequence attributes	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	The sequence number for this action. Mutually exclusive with Before, After, and OnExit attributes	

See Also

Wix Schema, CustomAction

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 <u>Custom Action Types</u> topic on MSDN.

Windows Installer references

CustomAction Table

Parents

Fragment, 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

Name	Туре	Description
BinaryKey	String	This attribute is a reference to with matching Id attribute. The contains the custom action of the custom action will not be target directory. This attribute with the DIIEntry attribute to action DLL to use for a type with the ExeCommand attribute 17 custom action that runs a executable, or with the VBSc JScriptCall attributes to spec custom action.
Directory	String	This attribute specifies a refe

		element with matching Id att directory path. This attribute with the ExeCommand attrib source executable for a type or with the Value attribute to string to place in the specifie entry in a type 35 custom ac
DIIEntry	String	This attribute specifies the na custom action to execute. used with the BinaryKey attribute 1 custom action, or with attribute to create a type 17
Error	String	This attribute specifies an inc table to use as an error mest custom action that displays t and aborts a product's instal
ExeCommand	String	This attribute specifies the constraints attribute specifies to supply to an executable. This attribute is the BinaryKey attribute for a action, the FileKey attribute action, the Property attribute custom action, or the Director type 34 custom action that specifies attribute to run.
Execute	Enumeration	This attribute indicates the scustom action. This attribute one of the following: commit Indicates that the custor after successful completinstallation script (at the installation).
		deferred Indicates that the custor script (possibly with elev

firstSequence

Indicates that the custor in the first sequence tha

immediate

Indicates that the custor during normal processin privileges. This is the de

oncePerProcess

Indicates that the custor in the first sequence tha process.

rollback

Indicates that a custom rollback sequence wher during installation, usua made by a deferred cus

secondSequence

Indicates that a custom a second time if it was p earlier sequence.

FileKey	String	This attribute specifies a reference element with matching Id attribute execute the custom action on the file is installed. This attribute with the ExeCommand attribute 18 custom action that runs a executable, with the DllEntry an installed custom action Dustom action, or with the JScriptCall attributes to specicustom action.
HideTarget	<u>YesNoType</u>	Ensures the installer does not CustomActionData for the deaction.

Id	String	The identifier of the custom a
Impersonate	YesNoType	This attribute specifies wheth Installer, which executes as should impersonate the user installing user when executir action. Typically the value sh except when the custom acti privileges to apply changes t
JScriptCall	String	This attribute specifies the natural function to execute in a scrip be provided in a Binary elem BinaryKey attribute describe words, this attribute must be conjunction with the BinaryK
PatchUninstall	YesNoType	This attribute specifies that t Installer, execute the custom patch is being uninstalled. The should also be conditioned u MSIPATCHREMOVE proper down level (less than Windows behavior.
Property	String	This attribute specifies a reference element with matching Id attribute Property to be used or up of this custom action. This at used with the Value attribute custom action that parses the places it into the specified Plattribute is also used with the attribute to create a type 50 uses the value of the given put the path to the executable. The actions are often useful to part deferred custom action. See http://msdn.microsoft.com/liktfor more information.

Return	Enumeration	Set this attribute to set the recustom action. This attribute one of the following: asyncNoWait Indicates that the custor asyncronously and execustor after the installer terminates.
		asyncWait Indicates that the custor asynchronously but the the return code at seque
		check Indicates that the custor synchronously and the r checked for success. The
		ignore Indicates that the custor synchronously and the r be checked.
Script	Enumeration	Creates a type 37 or 38 cust of the element should contai embedded in the package. T must be one of the following jscript vbscript
SuppressModularization	YesNoType	Use to suppress modularizat action name in merge modul be necessary for table-driver because the table name which cannot be modularized, so the one instance of the table.
TerminalServerAware	YesNoType	This attribute specifies controustom action will impersonauser during per-machine inst

		Server machines. Deferred eactions that do not specify the explicitly set it 'no', will run wimpersonation on Terminal Suring per-machine installationly applicable when installing Server 2003 family.
Value	String	This attribute specifies a strict the custom action. This attrik with the Property attribute to part of a type 51 custom action Directory attribute to set a distable in a type 35 custom action be a literal value or derived for element using the Formattec
VBScriptCall	String	This attribute specifies the name of Subroutine to execute in a subroutine subroutine to execute in a subroutine subro
Win64	YesNoType	Specifies that a script custor 64-bit platform. Valid only why Script, VBScriptCall, and JSc

Any Attribute (namespace='##other' processContents='lax') Extensik Schema. Schema extensions can register additional attributes at this

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, Module, PatchFamily, Product

Inner Text

None

Children

None

Attributes

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

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

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, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. <u>Column</u> (min: 0, max: unbounded): Column definition for the custom table.
- 2. Row (min: 0, max: unbounded): Row definition for the custom table.

Attributes

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

See Also

Wix Schema

Data Element

Description

Used for a Custom Table. Specifies the data for the parent Row and specified Column.

Windows Installer references

None

Parents

Row

Inner Text (xs:string)

A data value

Children

None

Attributes

Name	Туре	Description	Required
Column	String	Specifies in which column to insert this data.	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

DeleteServices Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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	Туре	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

Defines a dialog box in the Dialog Table.

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. <u>Control</u> (min: 0, max: unbounded): Control elements belonging to this dialog.

Name	Туре	Description	Required
Id	String	Unique identifier for the dialog.	Yes
CustomPalette	YesNoType	Used to specify if pictures in the dialog box are rendered with a custom palette.	
ErrorDialog	YesNoType	Specifies this dialog as an error dialog.	
Height	Integer	The height of the dialog box in dialog units.	Yes
Hidden	YesNoType	Used to hide the dialog.	

KeepModeless	YesNoType	Keep modeless dialogs alive when this dialog is created through DoAction.	
LeftScroll	YesNoType	Used to align the scroll bar on the left.	
Modeless	YesNoType	Used to set the dialog as modeless.	
NoMinimize	YesNoType	Used to specify if the dialog can be minimized.	
RightAligned	YesNoType	Align text on the right.	
RightToLeft	YesNoType	Used to specify if the text in the dialog should be displayed in right to left reading order.	
SystemModal	YesNoType	Used to set the dialog as system modal.	
Title	String	The title of the dialog box.	
TrackDiskSpace	YesNoType	Have the dialog periodically call the installer to check if available disk space has changed.	
Width	Integer	The width of the dialog box in dialog units.	Yes
X	Integer	Horizontal placement of the dialog box as a percentage of screen width. The default value is 50.	
Υ	Integer	Vertical placement of the	

dialog box as a percentage of screen height. The default value is 50.

See Also

Wix Schema

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, PackageCertificates, PatchCertificates

Inner Text

None

Children

None

Attributes

Name	Туре	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

None

Children

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

• <u>DigitalCertificate</u> (min: 1, max: 1)

Attributes

Name	Туре	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, Module, Product

Inner Text

None

Children

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

- Component (min: 0, max: unbounded)
- <u>Directory</u> (min: 0, max: unbounded)
- Merge (min: 0, max: unbounded)
- SymbolPath (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description
Id	String	This value is identifier of t
ComponentGuidGenerationSeed	Guid	The Compor Generation 5 must be used Component 9 guid directive rooted in a si

		Installer direct example, Proportion Commonfies recommer attribute be a developers in Components directories winstead (for embodies). It is note that one assigned a Component of Components directory nar
DiskId	Integer	Sets the defator for the files of directory. The may be over Component, or File element Merge element attribute for r
FileSource	String	Used to set t source for th elements. For information, source files.
LongName	LongFileNameType	This attribute deprecated;
LongSource	LongFileNameType	This attribute deprecated; SourceName

Name

LongFileNameType

The name of

Do not speci the LongNan directory rep directory as 1 the Windows <u>Directory tab</u> information a operator).

In prior versi toolset, this a the short dire attribute's va either a shor name. If a sh name is spec ShortName & be specified. name is spec LongName a be specified. is a long dire ShortName a omitted to all attempt to ge short directo However, if t with another wish to manu short directo ShortName & specified.

ShortName

<u>ShortFileNameType</u>

The short na directory in 8 attribute shouthere is a country of the short of the sh

		generated sh names or the manually spe directory nar
ShortSourceName	ShortFileNameType	The short na directory on in 8.3 format should only to conflict between short director user wants to the short souname.
SourceName	LongFileNameType	The name of the source m attribute is no Windows Ins to the Name
		In prior versitoolset, this athe short souname. This amay now be long directory directory narthe ShortSounattribute may If a long directory nare specified, the attribute may Also, if this voluments of the short source may be omitto attempt to unique short However, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be omitted attempt to unique short however, if the short source may be of the short source may be omitted attempt to unique short however, if the short source may be of the short source ma

		with another wish to manu short directo ShortSource may be spec
src	String	This attribute deprecated; FileSource a

How Tos and Examples

• How To: Add a file to your installer

See Also

Wix Schema, DirectoryRef

DirectoryRef Element

Description

Create a reference to a Directory element in another Fragment.

Windows Installer references

None

Parents

Fragment, Module, PatchFamily, 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)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description	Required
Id	String	The identifier of the Directory element to reference.	Yes
Diskld	Integer	Sets the default disk identifier for the files contained in this directory. This attribute's value may be overridden by a child Component, Directory, Merge or File element. See the File or Merge elements' DiskId attribute for more information.	

FileSource	String	Used to set the file system source for this DirectoryRef's child elements. For more information, see Specifying source files .
src	String	This attribute has been deprecated; please use the FileSource attribute instead.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

How Tos and Examples

• How To: Add a file to your installer

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

<u>ComplianceCheck, ComplianceDrive, ComponentSearch, DirectorySearch, DirectorySearch, Property, RegistrySearch</u>

Inner Text

None

Children

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

• <u>DirectorySearch</u> (min: 0, max: 1)

• <u>DirectorySearchRef</u> (min: 0, max: 1)

• FileSearch (min: 0, max: 1)

• FileSearchRef (min: 0, max: 1)

Name	Туре	Description	Required
Id	String	Unique identifier for the directory search.	Yes
AssignToProperty	YesNoType	Set the value of the outer Property to the result of this search. See remarks for more information.	
Depth	Integer	Depth below the path that the installer	

		searches for the file or directory specified by the search. See remarks for more information.
Path	String	Path on the user's system. Either absolute, or relative to containing directories.

Remarks

Use the AssignToProperty attribute to search for a file but set the outer property to the directory containing the file. When this attribute is set to 'yes', you may only nest a FileSearch element with a unique Id or define no child element.

When the parent DirectorySearch/@Depth attribute is greater than 1, the FileSearch/@Id attribute must be absent or the same as the parent DirectorySearch/@Id attribute value, unless the parent DirectorySearch/@AssignToProperty attribute value is 'yes'.

How Tos and Examples

- How To: Check the version number of a file during installation
- How To: Reference another DirectorySearch element
- How To: Get the parent directory of a file search

See Also

Wix Schema, ComponentSearch, IniFileSearch, RegistrySearch

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)

- <u>DirectorySearch</u> (min: 0, max: 1)
- <u>DirectorySearchRef</u> (min: 0, max: 1)
- FileSearch (min: 0, max: 1)
- FileSearchRef (min: 0, max: 1)

Name	Туре	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.

Remarks

A reference to another DirectorySearch element must reference the same Id, the same Parent Id, and the same Path. If any of these attribute values are different you must instead use a DirectorySearch element.

How Tos and Examples

• How To: Reference another DirectorySearch element

See Also

Wix Schema, ComponentSearch, IniFileSearch, RegistrySearch

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

DisableRollback Action

Parents

<u>InstallExecuteSequence</u>

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress	<u>YesNoType</u>	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

DuplicateFiles Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

EmbeddedChainer Element

Description

None

Windows Installer references

MsiEmbeddedChainer Table

Parents

Fragment, Module, Product

Inner Text (xs:string)

Element value is the condition. CDATA may be used to when a condition contains many XML characters that must be escaped. It is important to note that each EmbeddedChainer element must have a mutually exclusive condition to ensure that only one embedded chainer will execute at a time. If the conditions are not mutually exclusive the chainer that executes is undeterministic.

Children

None

Name	Type	Description	Required
Id	String	Unique identifier for embedded chainer.	Yes
BinarySource	String	Reference to the Binary element that contains the chainer executeable. Mutually exclusive with the FileSource and PropertySource attributes.	
CommandLine	String	Value to append to the transaction handle and passed to the chainer	

		executable.
FileSource	String	Reference to the File element that is the chainer executeable. Mutually exclusive with the BinarySource and PropertySource attributes.
PropertySource	String	Reference to a Property that resolves to the full path to the chainer executeable. Mutually exclusive with the BinarySource and FileSource attributes.

See Also

Wix Schema, Binary, File, Property, EmbeddedChainerRef

EmbeddedChainerRef Element

Description

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

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String		Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, EmbeddedChainer

EmbeddedUI Element

Description

Element value is the condition. Use CDATA if message contains delimiter characters.

Windows Installer references

MsiEmbeddedUI Table

Parents

UI

Inner Text (xs:string)

This element may have inner text.

Children

Sequence (min: 1, max: 1)

1. <u>EmbeddedUIResource</u> (min: 0, max: unbounded): Specifies extra files to be extracted for use by the embedded UI, such as language resources.

Name	Туре	Description
Id	String	Unique identifier for en attribute is not specifie attribute or the file nan SourceFile attribute wi
IgnoreActionData	<u>YesNoType</u>	Embedded UI will not r INSTALLLOGMODE_/ messages.
IgnoreActionStart	<u>YesNoType</u>	Embedded UI will not r INSTALLLOGMODE_/ messages.
IgnoreCommonData	YesNoType	Embedded UI will not r

		INSTALLLOGMODE_(messages.
IgnoreError	YesNoType	Embedded UI will not r INSTALLLOGMODE_E
IgnoreFatalExit	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreFilesInUse	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreInfo	<u>YesNoType</u>	Embedded UI will not r INSTALLLOGMODE_I
IgnoreInitialize	YesNoType	Embedded UI will not r INSTALLLOGMODE_I messages.
IgnoreInstallEnd	YesNoType	Embedded UI will not r INSTALLLOGMODE_I messages.
IgnoreInstallStart	YesNoType	Embedded UI will not r INSTALLLOGMODE_I messages.
IgnoreOutOfDiskSpace	YesNoType	Embedded UI will not r INSTALLLOGMODE_(messages.
IgnoreProgress	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreResolveSource	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreRMFilesInUse	<u>YesNoType</u>	Embedded UI will not r INSTALLLOGMODE_F

		messages.
IgnoreShowDialog	YesNoType	Embedded UI will not r INSTALLLOGMODE_S messages.
IgnoreTerminate	YesNoType	Embedded UI will not r INSTALLLOGMODE_1 messages.
IgnoreUser	<u>YesNoType</u>	Embedded UI will not r INSTALLLOGMODE_L
IgnoreWarning	YesNoType	Embedded UI will not r INSTALLLOGMODE_V messages.
Name	LongFileNameType	The name for the emberit is extracted from the executed. (Windows In support the typical sho filename combination files as it does for othe this attribute is not spe portion of the SourceFiused.
SourceFile	String	Path to the binary file t embedded UI. This mu exports the following the InitializeEmbeddedUI, EmbeddedUIHandler & ShutdownEmbeddedU
SupportBasicUI	YesNoType	Set yes to allow the Widisplay the embedded level installation.

See Also

EmbeddedUIResource Element

Description

Defines a resource for use by the embedded UI.

Windows Installer references

MsiEmbeddedUI Table

Parents

EmbeddedUI

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	Identifier for the embedded UI resource.	Yes
Name	LongFileNameType	The name for the resource when it is extracted from the Product for use by the embedded UI DLL. (Windows Installer does not support the typical short filename and long filename combination for embedded UI files as it does for other kinds of files.) If this attribute is not specified the Id	Yes

		attribute will be used.	
SourceFile	String	Path to the binary file that is the embedded UI resource.	Yes

See Also

Wix Schema, EmbeddedUI

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, Module, Product

Inner Text

None

Children

None

Attributes

Name	Туре	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

Environment Element

Description

Environment variables added or removed for the parent component.

Windows Installer references

Environment Table

Parents

Component

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	Unique identifier for environment entry.	Yes
Action	Enumeration	Specfies whether the environmental variable should be created, set or removed when the parent component is installed. This attribute's value must be one of the following: create 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 attributes. If you want to remove an environment variable, regardless of its value, do not set the Value attribute.

Name	String	Nam varia	ne of the environment able.	Yes
Part	Enumeration	one all	attribute's value must be of the following: This value is the entire environmental variable. This is the default.	
		first	This value is prefixed.	
		last	The second of production	

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 semicolon 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	The value to set into the environment variable. If this attribute is not set, the environment variable is removed during installation if it exists on the machine.

See Also

Wix Schema

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	Туре	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

Name	Type	Description	Requir
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

ExecuteAction Action

Parents

AdminUISequence, InstallUISequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

ExePackage Element

Description

Describes a single exe package to install.

Windows Installer references

None

Parents

Chain, PackageGroup

Inner Text

None

Children

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

- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema. The extension's CompilerExtension.ParseElement() method will be called with the package identifier as the first value in contextValues.

Name	Туре	Description	Required
After	String	The identifier of another package that this one should be installed after.	
Cache	<u>YesNoType</u>	Whether to cache the package.	
Cacheld	String	The identifier to use when caching the	

pac	kag	e.

		1	
DownloadUrl	String	The URL to use to download the package.	
Id	String	Identifier for this package, for ordering and cross-referencing.	
InstallCommand	String	The command-line arguments to specify to indicate an install.	
InstallCondition	String	A condition to evaluate before installing the package. The package will only be installed if the condition evaluates to true.	
Name	String	The destination path and file name for this chain payload. The default is the source file name. Use this attribute to rename the chain entry point or extract it into a subfolder.	
RepairCommand	String	The command-line arguments to specify to indicate a repair.	
SourceFile	String	Location of the package.	Yes
UninstallCommand	String	The command-line arguments to specify to indicate an	

un	ın	stall.	

Vital	 Whether the package should be considered "vital".	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema. The extension's CompilerExtension.ParseAttribute() method will be called with the package identifier in contextValues["PackageId"].

See Also

Wix Schema

Extension Element

Description

Extension for a Component

Windows Installer references

MIME Table, Verb Table, Registry Table

Parents

Component, Progld

Inner Text

None

Children

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

• MIME (min: 0, max: unbounded)

• Verb (min: 0, max: unbounded)

Attributes

Name	Туре	Description	
Id	String	This is simply the file extension, like Do not include the preceding period	
Advertise	YesNoType	Whether this extension is to be adv default is "no".	
ContentType	String	The MIME type that is to be written	
Any Attribute (namespace='##other' processContents='lax') Extensit Schema. Schema extensions can register additional attributes at this			
IsRichSavedGame		Registers this extension for the <u>rich</u> property handler on Windows Vista (http://schemas.microsoft.com/wix/	

See Also

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	Туре	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

Failure Element

Description

Failure action for a ServiceConfigFailureActions element.

Windows Installer references

None

Parents

<u>ServiceConfigFailureActions</u>

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Action	String	Specifies the action to take when the service fails. Valid values are "none", "restartComputer", "restartService", "runCommand" or a Formatted property that resolves to "0" (for "none"), "1" (for "restartService"), "2" (for "restartComputer") or "3" (for "runCommand").	Yes
Delay	String	Specifies the time in milliseconds to wait before performing the value from the Action attribute.	Yes

See Also

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. <u>UpgradeImage</u> (min: 1, max: unbounded)
- 2. Choice of elements (min: 0, max: unbounded)
 - ExternalFile (min: 0, max: unbounded)
 - ProtectFile (min: 0, max: unbounded)

Name	Туре	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

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, FeatureGroup, FeatureRef, Fragment, Product

Inner Text

None

Children

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

- Component (min: 0, max: unbounded)
- ComponentGroupRef (min: 0, max: unbounded)
- ComponentRef (min: 0, max: unbounded)
- Condition (min: 0, max: unbounded)
- Feature (min: 0, max: unbounded)
- FeatureGroupRef (min: 0, max: unbounded)
- FeatureRef (min: 0, max: unbounded)
- MergeRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description
Id	String	Unique identifier of the feature.
Absent	Enumeration	This attribute determines if a us

		option to set a feature to abser This attribute's value must be c allow Allows the user interface to change the feature state to disallow Prevents the user interface
		option to change the featu setting the msidbFeatureAttributesUII attribute. This will force the installation state, whether visible in the UI.
AllowAdvertise	Enumeration	This attribute determines the p states for this feature. This attr one of the following: no Prevents this feature from setting the msidbFeatureAttributesDis attribute.
		Prevents advertising for the operating system shell doe Windows Installer descript maidbFeatureAttributesNo attribute. yes Allows the feature to be ac
ConfigurableDirectory	String	Specify the Id of a Directory the by the user at installation time. be a public property and theref uppercase.
Description	String	Longer string of text describing

		localizable string is displayed b the Selection Dialog.
Display	String	Determines the initial display of feature tree. This attribute's value the following: collapse Initially shows the feature default value.
		expand Initially shows the feature
		hidden Prevents the feature from interface.
		<an explicit="" integer="" value=""> For advanced users only, i set the integer value of the appear in the Feature row.</an>
InstallDefault	Enumeration	This attribute determines the delocation of a feature. This attrib specified if the value of the Folloyes' since that would ask the infeature to follow the parent inst simultaneously favor a particular just for this feature. This attributione of the following: *followParent* Forces the feature to follow state as its parent feature.
		local Favors installing this featu the msidbFeatureAttributes
		source Favors running this feature setting the msidbFeatureA

attribute.

Level	Integer	Sets the install level of this feat disable the feature. Processing can modify the level value (this Condition child element).
Title	String	Short string of text identifying the is listed as an item by the Selection Dialog.
TypicalDefault	Enumeration	This attribute determines the dof the feature. This attribute's vithe following: advertise Sets the feature to be advertise Sets the feature Attributes Farthis value cannot be set if Allow Advertise attribute is ask the installer to disallow for this feature while at the install Sets the feature to the definition option.

Any Attribute (namespace='##other' processContents='lax') Extensik Schema. Schema extensions can register additional attributes at this

How Tos and Examples

• How To: Add a file to your installer

See Also

Wix Schema, FeatureRef

FeatureGroup Element

Description

Groups together multiple components, features, and merges to be used in other locations.

Windows Installer references

None

Parents

FeatureRef, Fragment

Inner Text

None

Children

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

- Component (min: 0, max: unbounded)
- <u>ComponentGroupRef</u> (min: 0, max: unbounded)
- ComponentRef (min: 0, max: unbounded)
- Feature (min: 0, max: unbounded)
- FeatureGroupRef (min: 0, max: unbounded)
- FeatureRef (min: 0, max: unbounded)
- MergeRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Attributes

Name	Type	Description	Required
Id	String	Identifier for the FeatureGroup.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, FeatureGroupRef

FeatureGroupRef Element

Description

Create a reference to a FeatureGroup in another Fragment.

Windows Installer references

None

Parents

Feature, FeatureGroup, FeatureRef, Product

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	The identifier of the FeatureGroup to reference.	Yes
IgnoreParent	YesNoType	Normally feature group references that end up nested under a parent element create a connection to that parent. This behavior is undesirable when trying to simply reference to a FeatureGroup in a different Fragment. Specify 'yes' to have this feature group reference not create a connection to its parent. The default is 'no'.	
Primary	<u>YesNoType</u>	Set this attribute to 'yes' in	

order to make the parent feature of this group the primary feature for any components and merges contained in the group. Features may belong to multiple features. By designating a feature as the primary feature of a component or merge, 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.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, FeatureGroup

FeatureRef Element

Description

Create a reference to a Feature element in another Fragment.

Windows Installer references

None

Parents

Feature, FeatureGroup, FeatureRef, Fragment, PatchFamily, Product

Inner Text

None

Children

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

- Component (min: 0, max: unbounded)
- <u>ComponentGroupRef</u> (min: 0, max: unbounded)
- ComponentRef (min: 0, max: unbounded)
- Feature (min: 0, max: unbounded)
- FeatureGroup (min: 0, max: unbounded)
- FeatureGroupRef (min: 0, max: unbounded)
- FeatureRef (min: 0, max: unbounded)
- MergeRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description	Required
Id	String	The identifier of the Feature element to reference.	Yes
IgnoreParent	YesNoType	Normally feature references	

that are nested under a parent element create a connection to that parent. This behavior is undesirable when trying to simply reference a Feature in a different Fragment. Specify 'yes' to have this feature reference not create a connection to its parent. The default is 'no'.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, Feature

File Element

Description

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

Windows Installer references

<u>File Table</u>

Parents

Component

Inner Text

None

Children

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

- Appld (min: 0, max: unbounded)
- AssemblyName (min: 0, max: unbounded)
- Class (min: 0, max: unbounded)
- <u>CopyFile</u> (min: 0, max: unbounded): Used to create a duplicate of this file elsewhere.
- ODBCDriver (min: 0, max: unbounded)
- ODBCTranslator (min: 0, max: unbounded)
- <u>Permission</u> (min: 0, max: unbounded): Used to configure the ACLs for this file.
- <u>PermissionEx</u> (min: 0, max: unbounded): Can also configure the ACLs for this file.
- Shortcut (min: 0, max: unbounded): Target of the shortcut will be set to this file.
- SymbolPath (min: 0, max: unbounded)
- TypeLib (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - EventManifest

- FirewallException
- FormatsFile
- Game
- HelpCollection
- HelpFile
- NativeImage
- PerfCounter
- <u>PerfCounterManifest</u>
- <u>PermissionEx</u>
- Snapln
- <u>TypesFile</u>

Name	Туре	Description
Assembly	Enumeration	Specifies if this File Win32 Assembly or .NET Assembly that needs to be installed into the Global Assembly Cache (G If the value is '.net' of 'win32', this file mus also be the key path the Component. Thi attribute's value mus one of the followings: .net The file is a .NE Framework assembly.
		no The file is not a .NET Framewo Win32 assembl This is the defa value.

win32

The file is a Wir assembly.

		assembly.
AssemblyApplication	String	Specifies the file identifier of the application file. This assembly will be isolated to the same directory as the application file. If thi attribute is absent, t assembly will be installed to the Glob Assembly Cache (G This attribute may o be specified if the Assembly attribute is to '.net' or 'win32'.
AssemblyManifest	String	Specifies the file identifier of the man file that describes th assembly. The mani file should be in the same component as assembly it describe This attribute may o be specified if the Assembly attribute is to '.net' or 'win32'.
BindPath	String	A list of paths, sepal by semicolons, that represent the paths be searched to find imported DLLs. The is usually a list of properties, with eacl

		property enclosed ir square brackets. Th value may be set to empty string. Includithis attribute will cau an entry to be gener for the file in the BindImage table.
Checksum	YesNoType	This attribute should set to "yes" for every executable file in the installation that has valid checksum stor the Portable Execut (PE) file header. On those files that have 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 depend on its own file versioning information but on the versionin its companion parer file that is the key pafor its component cannot be a companion (that means this attribute cannot be skeyPath="yes" for the file). The Version attribute cannot be set to make this set the set of

		along with this attrib since companion file are not installed bas on their own version
Compressed	<u>YesNoDefaultType</u>	Sets the file's source type compression. <i>A</i> setting of "yes" or "r will override the sett in the Word Count Summary Property.
DefaultLanguage	String	This is the default language of this file. linker will replace th value from the value the file if the supprefiles option is not us
DefaultSize	Integer	This is the default si of this file. The linke replace this value from the value in the file is suppress files option not used.
DefaultVersion	String	This is the default version of this file. T linker will replace th value from the value the file if the supprefiles option is not us
Diskld	Integer	The value of this attribute should correspond to the Id attribute of a Media element authored elsewhere. By creat this connection betwa file and its media,

		set the packaging options to the value specified in the Med element (values suc compression level, cembedding, etc). Specifying the Diskl attribute on the File element overrides the default Diskld attribute from the parent Component element or Diskld attribute is specified, the defaul "1". This Diskld attribute is ignored when creat a merge module because merge module do not have media.
FontTitle	String	Causes an entry to generated for the file the Font table with the specified FontTitle. attribute is intended be used to register the file as a non-TrueTy font.
Hidden	YesNoType	Set to yes in order to have the file's hidde attribute set when it installed on the targ machine.
Id	String	The unique identifie this File element. If your omit Id, it defaults to file name portion of Source attribute, if

		specified. May be referenced as a Property by specifyi [#value].
KeyPath	<u>YesNoType</u>	Set to yes in order to force this file to be to key path for the paracomponent.
LongName	<u>LongFileNameType</u>	This attribute has be deprecated; please the Name attribute instead.
Name	LongFileNameType	In prior versions of t WiX toolset, this attribute specified the short file name. This attribute's value may now be either a sho long file name. If a sefile name is specified the ShortName attribute a long file name is specified, the LongName attribute not be specified. Alse this value is a long finame, the ShortName attribute may be ome to allow WiX to attent to generate a unique short file name. However, if this name collides with anothe or you wish to manual specify the short file name, then the

		ShortName attribute may be specified. Finally, if this attribu omitted then its defavalue is the file nam portion of the Sourc attribute, if one is specified, or the valuatine Id attribute, if the Source attribute is omitted or doesn't contain a file name.
PatchAllowIgnoreOnError	<u>YesNoType</u>	Set to indicate that t patch is non-vital.
PatchGroup	Integer	This attribute must k set for patch-added Each patch should k assigned a different patch group number Patch groups number must be greater 0 at should be assigned consecutively. For example, the first passhould use PatchGroup='1', the second patch will have PatchGroup='2', etc
PatchIgnore	YesNoType	Prevents the updating the file that is in fact changed in the upgraded image related to the target images
PatchWholeFile	YesNoType	Set if the entire file should be installed rather than creating

		binary patch.
ProcessorArchitecture	Enumeration	Specifies the architecture for this assembly. This attrik should only be used .NET Framework 2.0 higher assemblies. attribute's value must one of the following: msil The file is a .NE Framework assembly that is processor-neutron x86 The file is a .NE Framework assembly for the x86 processor. x64 The file is a .NE Framework assembly for the x86 processor.
		ia64 The file is a .NE Framework assembly for th ia64 processor.
ReadOnly	YesNoType	Set to yes in order to have the file's readattribute set when it installed on the targmachine.
SelfRegCost	Integer	The cost of registeri

		the file in bytes. This must be a non-nega number. Including the attribute will cause a entry to be generate the file in the SelfRetable.
ShortName	ShortFileNameType	The short file name the file in 8.3 format This attribute should only be set if there is conflict between generated short file names or the user v to manually specify short file name.
Source	String	Specifies the path to File in the build produce Overrides default so path set by parent directories and Namattribute. This attribute must be set if no so information can be gathered from parendirectories. For more information, see Specifying source file.
src	String	This attribute has be deprecated; please the Source attribute instead.
System	YesNoType	Set to yes in order to have the file's syste attribute set when it installed on the targ

		machine.
TrueType	YesNoType	Causes an entry to generated for the file the Font table with r FontTitle specified. attribute is intended be used to register t file as a TrueType for
Vital	YesNoType	If a file is vital, then installation cannot proceed unless the is successfully insta. The user will have noption to ignore and installing this file. If error occurs, they camerely retry to instafile or abort the installation. The deficis "yes," unless the sfdvital switch (candle.exe) or SuppressFileDefaul property (.wixproj) is used.

Any Attribute (namespace='##other' processContents='lax') Extensik WiX XML Schema. Schema extensions can register additional attribuin the schema.

How Tos and Examples

• How To: Add a file to your installer

See Also

Wix Schema

FileCost Element

Description

Initiates dynamic costing of standard installation actions. Any standard or custom actions that affect costing should 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

FileCost Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>InstallExecuteSequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Name	Туре	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

FileSearch Element

Description

Searches for file and assigns to fullpath value of parent Property

Windows Installer references

DrLocator Table, Signature Table

Parents

ComponentSearch, DirectorySearch, DirectorySearchRef, IniFileSearch, RegistrySearch

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	Unique identifier for the file search and external key into the Signature table. If this attribute value is not set then the parent element's @Id attribute is used.	
Languages	String	The languages supported by the file.	
LongName	LongFileNameType	This attribute has been deprecated; please use the Name attribute instead.	

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	<u>LongFileNameType</u>	In prior versions of the WiX toolset, this attribute specified the short file name. This attribute's value may	

now be either a short or long file name. If a short file name is specified, the ShortName attribute may not be specified. If a long file name is specified, the LongName attribute may not be specified. If you wish to manually specify the short file name, then the ShortName attribute may be specified.

ShortName <u>ShortFileNameType</u>

The short file name of the file in 8.3 format. There is a Windows Installer bug which prevents the FileSearch functionality from working if both a short and long file name are specified. Since the Name attribute allows either a short or long name to be specified, it is the only attribute related to file names which should be specified.

Remarks

When the parent DirectorySearch/@Depth attribute is greater than 1, the FileSearch/@Id attribute must be absent or the same as the

parent DirectorySearch/@Id attribute value, unless the parent DirectorySearch/@AssignToProperty attribute value is 'yes'.

How Tos and Examples

• How To: Check the version number of a file during installation

See Also

<u>Wix Schema, ComponentSearch, DirectorySearch, DirectorySearch, PileSearchRef, IniFileSearch, RegistrySearch</u>

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	Туре	Description	Required
Id	String	Specify the Id to the FileSearch to reference.	Yes

Remarks

A reference to another FileSearch element must reference the same Id and the same Parent Id. If any of these attribute values are different you must instead use a FileSearch element.

See Also

Wix Schema, FileSearch

FileTypeMask Element

Description

FileType data for class Id registration.

Windows Installer references

None

Parents

Class

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Mask	<u>HexType</u>	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

FindRelatedProducts Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

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.
Overridable	<u>YesNoType</u>	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.
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

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

ForceReboot Action

Parents

<u>InstallExecuteSequence</u>

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress	<u>YesNoType</u>	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

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

- Appld (min: 0, max: unbounded)
- Binary (min: 0, max: unbounded)
- ComplianceCheck (min: 0, max: unbounded)
- Component (min: 0, max: unbounded)
- ComponentGroup (min: 0, max: unbounded)
- Condition (min: 0, max: unbounded)
- Container (min: 0, max: unbounded)
- <u>CustomAction</u> (min: 0, max: unbounded)
- CustomActionRef (min: 0, max: unbounded)

- <u>CustomTable</u> (min: 0, max: unbounded)
- Directory (min: 0, max: unbounded)
- <u>DirectoryRef</u> (min: 0, max: unbounded)
- EmbeddedChainer (min: 0, max: unbounded)
- EmbeddedChainerRef (min: 0, max: unbounded)
- EnsureTable (min: 0, max: unbounded)
- Feature (min: 0, max: unbounded)
- FeatureGroup (min: 0, max: unbounded)
- FeatureRef (min: 0, max: unbounded)
- Icon (min: 0, max: unbounded)
- <u>IgnoreModularization</u> (min: 0, max: unbounded)
- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- Media (min: 0, max: unbounded)
- PackageCertificates (min: 0, max: unbounded)
- PackageGroup (min: 0, max: unbounded)
- PatchCertificates (min: 0, max: unbounded)
- PatchFamily (min: 0, max: unbounded)
- PayloadGroup (min: 0, max: unbounded)
- Property (min: 0, max: unbounded)
- PropertyRef (min: 0, max: unbounded)
- <u>SetDirectory</u> (min: 0, max: unbounded)
- <u>SetProperty</u> (min: 0, max: unbounded)
- SFPCatalog (min: 0, max: unbounded)
- <u>UI</u> (min: 0, max: unbounded)
- <u>UIRef</u> (min: 0, max: unbounded)
- Upgrade (min: 0, max: unbounded)
- WixVariable (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 - 1. <u>InstallExecuteSequence</u> (min: 0, max: 1)
 - 2. <u>InstallUISequence</u> (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') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - CloseApplication
 - ComPlusApplication
 - ComPlusApplicationRole
 - ComPlusPartition
 - ComPlusPartitionRole
 - ComponentSearch
 - DirectorySearch
 - FileSearch
 - Group
 - HelpCollectionRef
 - HelpFilter
 - LayoutDirectory
 - LayoutDirectoryRef
 - Mutation
 - PackageGroup
 - PayloadGroup
 - ProductSearch
 - RegistrySearch
 - SqlDatabase
 - UnitTest
 - UnitTestRef
 - User
 - UX
 - WebApplication
 - WebAppPool
 - WebDirProperties
 - WebLog
 - WebSite

Attributes

Name	Туре	Description	Required
Id	String	Optional identifier for a Fragment. Should only be set by advanced users to tag sections.	

See Also

Wix Schema

Icon Element

Description

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

Windows Installer references

Icon Table

Parents

Fragment, Module, Product, Shortcut

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The Id cannot by 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.	

How Tos and Examples

- How To: Set your installer's icon in Add/Remove Programs
- How To: Create a shortcut on the Start Menu

See Also

Wix Schema

IconRef Element

Description

Used only for PatchFamilies to include only a icon table entry in a patch.

Windows Installer references

None

Parents

PatchFamily

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The identifier of the Icon element to reference.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema

IgnoreModularization Element

Description

This element has been deprecated. Use the Binary/@SuppressModularization, CustomAction/@SuppressModularization, or Property/@SuppressModularization attributes instead.

Windows Installer references

None

Parents

Fragment, Module

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Name	String	The name of the item to ignore modularization for.	Yes
Type	Enumeration	The type of the item to ignore modularization for. This attribute's value must be one of the following: <i>Action</i>	
		Property	
		Directory	

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	Туре	Description	Required
Length	Int	Length of the range.	Yes
Offset	Int	Offset of the start of the range.	Yes

See Also

Wix Schema

IgnoreTable Element

Description

Specifies a table from the merge module that is not merged into an .msi file. If the table already exists in an .msi file, it is not modified by the merge. The specified table can therefore contain data that is unneeded after the merge. To minimize the size of the .msm file, it is recommended that developers remove unused tables from modules intended for redistribution rather than creating IgnoreTable elements for those tables.

Windows Installer references

None

Parents

Module

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
ld	String	The name of the table in the merge module that is not to be merged into the .msi file.	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)

Any Element (namespace='##any' processContents='Lax')

Attributes

None

See Also

Wix Schema

IniFile Element

Description

Adds or removes .ini file entries.

Windows Installer references

IniFile Table, RemoveIniFile Table

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Identifier for ini file.	Yes
Action	Enumeration	The type of modification to be made. This attribute's value must be one of the following: addLine Creates or updates an .ini entry.	Yes
		addTag Creates a new entry or appends a new comma- separated value to an existing	

entry.

createLine

Creates an .ini entry only if the entry does 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	<u>LongFileNameType</u>	This attribute has been deprecated; please use the Name attribute instead.	
Name	LongFileNameType	In prior versions of the WiX toolset, this attribute specified the	Yes

short name. This attribute's value may now be either a short or long name. If a short name is specified, the ShortName attribute may not be specified. If a long name is specified, the LongName attribute may not be specified. Also, if this value is a long name, the ShortName attribute may be omitted to allow WiX to attempt to generate a unique short name. However, if this name collides with another file or you wish to manually specify the short name, then the ShortName attribute may be specified.

Section	String	The localizable .ini file section.	Yes
ShortName	ShortFileNameType	The short name of the in 8.3 format. This attribute should only be set if there is a conflict between generated short names or the user wants to manually specify the short name.	

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

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)

- <u>DirectorySearch</u> (min: 0, max: 1)
- <u>DirectorySearchRef</u> (min: 0, max: 1)
- FileSearch (min: 0, max: 1)
- FileSearchRef (min: 0, max: 1)

Attributes

Name	Туре	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	This attribute has been deprecated;	

		please use the Name attribute instead.	
Name	LongFileNameType	In prior versions of the WiX toolset, this attribute specified the short name. This attribute's value may now be either a short or long name. If a short name is specified, the ShortName attribute may not be specified. If a long name is specified, the LongName attribute may not be specified. Also, if this value is a long name, the ShortName attribute may be omitted to allow WiX to attempt to generate a unique short name. However, if you wish to manually specify the short name, then the ShortName attribute may be specified.	Yes
Section	String	The localizable .ini file section.	Yes
ShortName	ShortFileNameType	The short name of the file in 8.3 format. This attribute should only be set if the user wants to manually	

specify	the	short
name.		

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 must

be one of the following: directory

A directory location.

file

A file location. This is the default value.

raw

A raw .ini value.

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

InstallAdminPackage Action

Parents

AdminExecuteSequence, AdminUISequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

InstallExecute Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
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

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

InstallExecuteAgain Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the	

		position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema

InstallExecuteSequence Element

Description

None

Windows Installer references

<u>InstallExecuteSequence Table</u>

Parents

Fragment, Module, Product

Inner Text

None

Children

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

- <u>AllocateRegistrySpace</u> (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.
- <u>BindImage</u> (min: 0, max: unbounded): Binds each executable or DLL that must be bound to the DLLs imported by it.
- <u>CCPSearch</u> (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- <u>CostFinalize</u> (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- <u>CostInitialize</u> (min: 0, max: unbounded): Initiates the internal installation costing process.
- <u>CreateFolders</u> (min: 0, max: unbounded): Creates empty folders for components that are set to be installed.
- <u>CreateShortcuts</u> (min: 0, max: unbounded): Manages the creation of shortcuts.
- <u>Custom</u> (min: 0, max: unbounded): Use to sequence a custom action.

- <u>DeleteServices</u> (min: 0, max: unbounded): Stops a service and removes its registration from the system.
- <u>DisableRollback</u> (min: 0, max: unbounded): Disables rollback for the remainder of the installation.
- <u>DuplicateFiles</u> (min: 0, max: unbounded): Duplicates files installed by the InstallFiles action.
- FileCost (min: 0, max: unbounded): Initiates dynamic costing of standard installation actions.
- <u>FindRelatedProducts</u> (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.
- <u>InstallExecuteAgain</u> (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.
- <u>InstallFinalize</u> (min: 0, max: unbounded): Marks the end of a sequence of actions that change the system.
- <u>InstallInitialize</u> (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.
- <u>InstallServices</u> (min: 0, max: unbounded): Registers a service for the system.
- <u>InstallValidate</u> (min: 0, max: unbounded): Verifies that all costed volumes have enough space for the installation.

- <u>IsolateComponents</u> (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).
- <u>LaunchConditions</u> (min: 0, max: unbounded): Queries the LaunchCondition table and evaluates each conditional statement recorded there.
- <u>MigrateFeatureStates</u> (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.
- <u>MsiPublishAssemblies</u> (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.
- <u>PatchFiles</u> (min: 0, max: unbounded): Queries the Patch table to determine which patches are to be applied.
- <u>ProcessComponents</u> (min: 0, max: unbounded): Registers and unregisters components, their key paths, and the component clients.
- <u>PublishComponents</u> (min: 0, max: unbounded): Manages the advertisement of the components from the PublishComponent table.
- <u>PublishFeatures</u> (min: 0, max: unbounded): Writes each feature's state into the system registry.
- <u>PublishProduct</u> (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.
- <u>RegisterComPlus</u> (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.
- <u>RegisterMIMEInfo</u> (min: 0, max: unbounded): Registers MIME-

- related registry information with the system.
- <u>RegisterProduct</u> (min: 0, max: unbounded): Registers the product information with the installer.
- RegisterProgldInfo (min: 0, max: unbounded): Manages the registration of OLE Progld 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.
- RemovelniValues (min: 0, max: unbounded): Removes .ini file information specified for removal in the RemovelniFile 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.
- <u>RemoveRegistryValues</u> (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.
- <u>ScheduleReboot</u> (min: 0, max: unbounded): Prompts the user to restart the system at the end of installation. Not fixed sequence.
- <u>SelfRegModules</u> (min: 0, max: unbounded): Processes all modules listed in the SelfReg table and registers all installed modules with the system.
- <u>SelfUnregModules</u> (min: 0, max: unbounded): Unregisters all modules listed in the SelfReg table that are scheduled to be uninstalled.
- <u>SetODBCFolders</u> (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.
- <u>UnpublishComponents</u> (min: 0, max: unbounded): Manages the unadvertisement of components listed in the PublishComponent table.
- <u>UnpublishFeatures</u> (min: 0, max: unbounded): Removes selection-state and feature-component mapping information from the registry.
- <u>UnregisterClassInfo</u> (min: 0, max: unbounded): Manages the removal of COM class information from the system registry.
- <u>UnregisterComPlus</u> (min: 0, max: unbounded): Removes COM+ applications from the registry.
- <u>UnregisterExtensionInfo</u> (min: 0, max: unbounded): Manages the removal of extension-related information from the system registry.
- <u>UnregisterFonts</u> (min: 0, max: unbounded): Removes registration information about installed fonts from the system.
- <u>UnregisterMIMEInfo</u> (min: 0, max: unbounded): Unregisters MIME-related registry information from the system.
- <u>UnregisterProgIdInfo</u> (min: 0, max: unbounded): Manages the

- unregistration of OLE Progld information with the system.
- <u>UnregisterTypeLibraries</u> (min: 0, max: unbounded): Unregisters type libraries from the system.
- <u>ValidateProductID</u> (min: 0, max: unbounded): Sets the ProductID property to the full product identifier.
- <u>WriteEnvironmentStrings</u> (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.
- <u>WriteRegistryValues</u> (min: 0, max: unbounded): Sets up an application's registry information.

Attributes

None

See Also

Wix Schema

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

InstallFiles Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

InstallFinalize Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

InstallInitialize Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

InstallODBC Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

InstallServices Action

Parents

<u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

InstallUISequence Table

Parents

Fragment, Module, Product, UI

Inner Text

None

Children

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

- <u>AppSearch</u> (min: 0, max: unbounded): Uses file signatures to search for existing versions of products.
- <u>CCPSearch</u> (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- <u>CostFinalize</u> (min: 0, max: unbounded): Ends the internal installation costing process begun by the CostInitialize action.
- <u>CostInitialize</u> (min: 0, max: unbounded): Initiates the internal installation costing process.
- <u>Custom</u> (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.
- <u>FindRelatedProducts</u> (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.

- <u>IsolateComponents</u> (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).
- <u>LaunchConditions</u> (min: 0, max: unbounded): Queries the LaunchCondition table and evaluates each conditional statement recorded there.
- <u>MigrateFeatureStates</u> (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.
- <u>ScheduleReboot</u> (min: 0, max: unbounded): Prompts the user to restart the system at the end of installation. Not fixed sequence.
- Show (min: 0, max: unbounded): Displays a Dialog.
- <u>ValidateProductID</u> (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

InstallValidate Action

Parents

<u>AdminExecuteSequence</u>, <u>AdminUISequence</u>, <u>AdvertiseExecuteSequence</u>, <u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Instance Element

Description

Defines an instance transform for your product.

Windows Installer references

None

Parents

InstanceTransforms

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	The identity of the instance transform. This value will define the name by which the instance should be referred to on the command line. In addition, the value of the this attribute will determine what the value of the property specified in Property attribute on InstanceTransforms will change to for each instance.	Yes
ProductCode	String	The ProductCode for this instance.	Yes
ProductName	String	The ProductName for this instance.	

See Also Wix Schema

InstanceTransforms Element

Description

Use this element to contain definitions for instance transforms.

Windows Installer references

None

Parents

Product

Inner Text

None

Children

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

• Instance (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Property	String	The Id of the Property who's value should change for each instance.	Yes

See Also

Wix Schema

Interface Element

Description

COM Interface registration for parent TypeLib.

Windows Installer references

Registry Table

Parents

Class, Component, TypeLib

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	Guid	GUID identifier for COM Interface.	Yes
BaseInterface	Guid	Identifies the interface from which the current interface is derived.	
Name	String	Name for COM Interface.	Yes
NumMethods	Integer	Number of methods implemented on COM Interface.	
ProxyStubClassId	Guid	GUID CLSID for proxy stub to COM Interface.	

ProxyStubClassId32	Guid	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

IsolateComponent Element

Description

Shared Component to be privately replicated in folder of parent Component

Windows Installer references

IsolateComponent Table

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	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

IsolateComponents Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

This element may have inner text.

Children

None

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

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

LaunchConditions Action

Parents

AdminExecuteSequence, AdminUISequence, InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of	

		this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema, Condition

LayoutDirectory Element

Description

Representation of layout directory.

Windows Installer references

None

Parents

Bundle, Fragment, LayoutDirectory, LayoutDirectoryRef, Module, Patch, PatchCreation, Product

Inner Text

None

Children

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

- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- LayoutFile (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	String	Identifier for LayoutDirectory. Only required if this is shared and is referenced by LayoutDirectoryRef.	
Name	String	The destination path of this layout. It can contain multiple levels of directory structure (i.e. src\files\product)	Yes

See Also

Wix Schema

LayoutDirectoryRef Element

Description

Create a reference to an existing LayoutDirectory element.

Windows Installer references

None

Parents

Bundle, Fragment, LayoutDirectory, LayoutDirectoryRef, Module, Patch, PatchCreation, Product

Inner Text

None

Children

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

- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- LayoutFile (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	String	The identifier of LayoutDirectory element to reference.	Yes

See Also

Wix Schema, LayoutDirectory

LayoutFile Element

Description

Describes the payload to a bootstrapper.

Windows Installer references

None

Parents

LayoutDirectory, LayoutDirectoryRef

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Name	String	The destination path and file name for this file. The default is the source file name.	
SourceFile	String	Location of the source file.	Yes

See Also

Wix Schema

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	Туре	Description	Required
Property	String	Property tied to this group	Yes

See Also

Wix Schema

ListItem Element

Description

The value (and optional text) associated with an item in a ComboBox, ListBox, or ListView.

Windows Installer references

ComboBox Table, ListBox Table, ListView Table

Parents

ComboBox, ListBox, ListView

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Icon	String	The identifier of the Binary (not Icon) element containing the icon to associate with this item. This value is only valid when nested under a ListView element.	
Text	String	The localizable, visible text to be assigned to the item. If not specified, this will default to the value of the Value attribute.	
Value	String	The value assigned to the associated ComboBox, ListBox, or ListView property if this item is selected.	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	Туре	Description	Required
Property	String	Property tied to this group	Yes

See Also

Wix Schema

MajorUpgrade Element

Description

Simplifies authoring for major upgrades, including support for preventing downgrades.

The parent Product element must have valid UpgradeCode and Version attributes.

Windows Installer references

None

Parents

Product

Inner Text

None

Children

None

Name	Туре	Description
AllowDowngrades	YesNoType	When set to no (the default products with lower versior numbers are blocked from installing when a product wa higher version is installed the DowngradeErrorMessa attribute must also be specified. When set to yes, any version is installed over any other version.
DowngradeErrorMessage	String	The message displayed if

		users try to install a produc with a lower version numbe when a product with a high version is installed. Used o when AllowDowngrades is (the default).
IgnoreRemoveFailure	YesNoType	When set to yes, failures removing the installed product during the upgrade will be ignored.
		When set to no (the default failures removing the installed product during the upgrade will be considered failure and, depending on t scheduling, roll back the upgrade.
MigrateFeatures	YesNoType	When set to yes (the defau the MigrateFeatureStates standard action will set the feature states of the upgrat product to those of the installed product.
		When set to no, the installe features have no effect on upgrade installation.
RemoveFeatures	String	A formatted string that contains the list of features remove from the installed product. The default is to remove all features. Note the if you use formatted proper values that evaluate to an empty string, no features where the temoved; only omitting

this attribute defaults to removing all features.

Schedule

Enumeration Determines the scheduling the RemoveExistingProduc standard action, which is when the installed product removed. The default is "afterInstallValidate" which removes the installed produ entirely before installing the upgrade product. It's slowe but gives the most flexibility in changing components ar features in the upgrade product.

> For more information, see RemoveExistingProducts. This attribute's value must one of the following: afterInstallValidate

> > (Default) Schedules RemoveExistingProdu after the InstallValidate standard action. This scheduling removes th installed product entire before installing the upgrade product. It's slowest but gives the most flexibility in changing components and features in the upgrade product. Note that if the installation o the upgrade product fails, the machine will have neither version

installed.

afterInstallInitialize

Schedules
RemoveExistingProdu
after the InstallInitialize
standard action. This is
similar to the
afterInstallValidate
scheduling, but if the
installation of the
upgrade product fails,
Windows Installer also
rolls back the removal
the installed product -other words, reinstalls

afterInstallExecute

Schedules RemoveExistingProdu between the InstallExecute and InstallFinalize standard actions. This schedulir installs the upgrade product "on top of" the installed product then lets RemoveExistingProdu uninstall any components that don't also exist in the upgrad product. Note that this scheduling requires str adherence to the component rules because it relies on component reference counts to be accurate during installation of th

upgrade product and removal of the installed product. For more information, see Bob Arnson's blog post "Paying for Upgrades" for details. If installatio of the upgrade product fails, Windows Installed also rolls back the removal of the installed product -- in other work reinstalls it.

afterInstallExecuteAgain

Schedules
RemoveExistingProdu
between the
InstallExecuteAgain ar
InstallFinalize standard
actions. This is identicated to the afterInstallExecuteAgain
scheduling but after the
InstallExecuteAgain
standard action instead
of InstallExecute.

afterInstallFinalize

Schedules
RemoveExistingProdu
after the InstallFinalize
standard action. This is
similar to the
afterInstallExecute and
afterInstallExecuteAga
schedulings but takes
place outside the
installation transaction
so if installation of the
upgrade product fails,

Windows Installer does not roll back the remove of the installed product so the machine will have both versions installed

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, Patch, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - <u>DigitalSignature</u> (min: 0, max: unbounded)
 - PatchBaseline (min: 0, max: unbounded)
 - SymbolPath (min: 0, max: unbounded)

Name	Туре	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 in a cabinet file. If no cabinets are used, this attribute must not	

be set.	

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 must 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).
Source	String	Optional property that identifies the source of the embedded cabinet. If a cabinet is specified for a patch, this property should be defined and unique

to each patch so that the embedded cabinet containing patched and new files can be located in the patch package. If the cabinet is not embedded - this is not typical - the cabinet can be found in the directory referenced in this column. If empty, the external cabinet must be located in the SourceDir directory. String This attribute has src 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

Merge Element

Description

Merge directive to bring in a merge module that will be redirected to the parent directory.

Windows Installer references

None

Parents

Directory, **DirectoryRef**

Inner Text

None

Children

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

• <u>ConfigurationData</u> (min: 0, max: unbounded): Data to use as input to a configurable merge module.

Name	Туре	Description	Required
Id	String	The unique identifier for the Merge element in the source code. Referenced by the MergeRef/@Id.	Yes
Diskld	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 as compression level, cab embedding, etc).	
FileCompression	<u>YesNoType</u>	Specifies if the files in the merge module should be compressed.	
Language	LocalizableInteger	Specifies the decimal LCID or localization token for the language to merge the Module in as.	Yes
SourceFile	String	Path to the source location of the merge module.	
SrC	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

How Tos and Examples

• How To: Install the Visual C++ Redistributable with your installer

See Also

Wix Schema, MergeRef

MergeRef Element

Description

Merge reference to connect a Merge Module to parent Feature

Windows Installer references

None

Parents

Feature, FeatureGroup, FeatureRef

Inner Text

None

Children

None

Attributes

Name	Туре	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.	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

How Tos and Examples

• How To: Install the Visual C++ Redistributable with your installer

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

MigrateFeatureStates Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Name	Туре	Description	Required
Advertise	YesNoType	Whether this MIME is to be advertised. The default is to match whatever the parent extension element uses. If the parent element is not advertised, then this element cannot be advertised either.	
Class	Guid	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

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

<u>Wix</u>

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Package (min: 1, max: 1)
- 2. Choice of elements (min: 0, max: unbounded)
 - Appld (min: 0, max: unbounded)
 - Binary (min: 0, max: unbounded)
 - Component (min: 0, max: unbounded)
 - <u>ComponentGroupRef</u> (min: 0, max: unbounded)
 - ComponentRef (min: 0, max: unbounded)
 - Configuration (min: 0, max: unbounded)
 - CustomAction (min: 0, max: unbounded)
 - <u>CustomActionRef</u> (min: 0, max: unbounded)
 - <u>CustomTable</u> (min: 0, max: unbounded)
 - <u>Dependency</u> (min: 0, max: unbounded)
 - <u>Directory</u> (min: 0, max: unbounded)
 - <u>DirectoryRef</u> (min: 0, max: unbounded)
 - EmbeddedChainer (min: 0, max: unbounded)

- EmbeddedChainerRef (min: 0, max: unbounded)
- EnsureTable (min: 0, max: unbounded)
- Exclusion (min: 0, max: unbounded)
- <u>Icon</u> (min: 0, max: unbounded)
- <u>IgnoreModularization</u> (min: 0, max: unbounded)
- IgnoreTable (min: 0, max: unbounded)
- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- Property (min: 0, max: unbounded)
- <u>PropertyRef</u> (min: 0, max: unbounded)
- <u>SetDirectory</u> (min: 0, max: unbounded)
- <u>SetProperty</u> (min: 0, max: unbounded)
- <u>SFPCatalog</u> (min: 0, max: unbounded)
- <u>Substitution</u> (min: 0, max: unbounded)
- UI (min: 0, max: unbounded)
- <u>UIRef</u> (min: 0, max: unbounded)
- WixVariable (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 - 1. <u>InstallExecuteSequence</u> (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')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - CloseApplication
 - ComPlusApplication
 - ComPlusApplicationRole
 - ComPlusPartition
 - ComPlusPartitionRole
 - Group

- HelpCollectionRef
- <u>HelpFilter</u>
- LayoutDirectory
- LayoutDirectoryRef
- SqlDatabase
- User
- WebApplication
- WebAppPool
- WebDirProperties
- WebLog
- WebSite

Attributes

Name	Туре	Description	Required
Id	String	The name of the merge module (not the file name).	Yes
Codepage	String	The code page integer value or web name for the resulting MSM. See remarks for more information.	
Guid	Guid	This attribute is deprecated. Use the Package/@Id attribute instead.	
Language	LocalizableInteger	The decimal language ID (LCID) of the merge module.	Yes
Version	String	The major and minor versions of the merge module.	Yes

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

See Also

Wix Schema

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

MoveFile Table, MoveFiles Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

MsiPackage Element

Description

Describes a single msi package to install.

Windows Installer references

None

Parents

Chain, PackageGroup

Inner Text

None

Children

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

- MsiProperty (min: 0, max: unbounded)
- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema. The extension's CompilerExtension.ParseElement() method will be called with the package identifier as the first value in contextValues.
 - MsiProperty

Name	Туре	Description	Required
After	String	The identifier of another package that this one should be installed after.	
Cache	YesNoType	Whether to cache the package.	

Cacheld	String	The identifier to use when caching the package.	
DownloadUrl	String	The URL to use to download the package.	
Id	String	Identifier for this package, for ordering and cross-referencing.	
InstallCondition	String	A condition to evaluate before installing the package. The package will only be installed if the condition evaluates to true.	
Name	String	The destination path and file name for this chain payload. The default is the source file name. Use this attribute to rename the chain entry point or extract it into a subfolder.	
SourceFile	String	Location of the package.	Yes
Vital	YesNoType	Whether the package should be considered "vital".	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema. The extension's CompilerExtension.ParseAttribute() method will be called with the package identifier in contextValues["PackageId"].

See Also

Wix Schema

MsiProperty Element

Description

Allows an MSI property to be set based on the value of a burn engine expression.

Windows Installer references

None

Parents

MsiPackage

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Name	String	The name of the MSI property to set.	Yes
Value	String	The value to set the property to. This string is evaluated by the burn engine and can be as simple as a burn engine variable reference or as complex as a full expression.	Yes

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

MsiPublishAssemblies Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

MsiUnpublishAssemblies Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

Wix Schema

MspPackage Element

Description

Describes a single msp package to install.

Windows Installer references

None

Parents

Chain, PackageGroup

Inner Text

None

Children

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

- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema. The extension's CompilerExtension.ParseElement() method will be called with the package identifier as the first value in contextValues.

Name	Туре	Description	Required
After	String	The identifier of another package that this one should be installed after.	
Cache	YesNoType	Whether to cache the package.	
Cacheld	String	The identifier to use when caching the package.	

DownloadUrl	String	The URL to use to download the package.	
Id	String	Identifier for this package, for ordering and cross-referencing.	
InstallCondition	String	A condition to evaluate before installing the package. The package will only be installed if the condition evaluates to true.	
Name	String	The destination path and file name for this chain payload. The default is the source file name. Use this attribute to rename the chain entry point or extract it into a subfolder.	
SourceFile	String	Location of the package.	Yes
Vital	YesNoType	Whether the package should be considered "vital".	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema. The extension's CompilerExtension.ParseAttribute() method will be called with the package identifier in contextValues["PackageId"].

See Also

Wix Schema

MsuPackage Element

Description

Describes a single msu package to install.

Windows Installer references

None

Parents

Chain, PackageGroup

Inner Text

None

Children

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

- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema. The extension's CompilerExtension.ParseElement() method will be called with the package identifier as the first value in contextValues.

Name	Туре	Description	Required
After	String	The identifier of another package that this one should be installed after.	
Cache	<u>YesNoType</u>	Whether to cache the package.	
Cacheld	String	The identifier to use when caching the package.	

DownloadUrl	String	The URL to use to download the package.	
Id	String	Identifier for this package, for ordering and cross-referencing.	
InstallCondition	String	A condition to evaluate before installing the package. The package will only be installed if the condition evaluates to true.	
Name	String	The destination path and file name for this chain payload. The default is the source file name. Use this attribute to rename the chain entry point or extract it into a subfolder.	
SourceFile	String	Location of the package.	Yes
Vital	YesNoType	Whether the package should be considered "vital".	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema. The extension's CompilerExtension.ParseAttribute() method will be called with the package identifier in contextValues["PackageId"].

See Also

Wix Schema

MultiStringValue 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 MultiStringValue element.

Windows Installer references

Registry Table

Parents

RegistryValue

See Also

Wix Schema

ODBCDataSource Element

Description

ODBCDataSource for a Component

Windows Installer references

ODBCDataSource Table

Parents

Component, ODBCDriver

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Name	Туре	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 must be one of the following:	Yes

machine

Data source is registered per machine.

user

Data source is registered per user.

See Also

Wix Schema

ODBCDriver Element

Description

ODBCDriver for a Component

Windows Installer references

ODBCDriver Table

Parents

Component, File

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	Туре	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

Inner Text

None

Children

None

Attributes

Name	Туре	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

OptimizeCustomActions Element

Description

Indicates whether custom actions can be skipped when applying the patch.

Windows Installer references

MsiPatchMetadata Table

Parents

Patch, PatchMetadata

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
SkipAssignment	<u>YesNoType</u>	Skip property (type 51) and directory (type 35) assignment custom actions.	
SkipDeferred	YesNoType	Skip custom actions that run within the script.	
SkipImmediate	YesNoType	Skip immediate custom actions that are not property or directory assignment custom actions.	

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

Name	Туре	Description	Requ
AdminImage	<u>YesNoType</u>	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. This attribute cannot be set for merge modules.	
Description	String	The product full name or	

		description.
Id	AutogenGuid	The package code GUID for a product or merge module. When compiling a product, this attribute should not be set in order to allow the package code to be generated for each build. When compiling a merge module, this attribute must be set to the modularization guid.
InstallerVersion	Integer	The minimum version of the Windows Installer required to install this package. Take the major version of the required Windows Installer and multiply by a 100 then add the minor version of the Windows Installer. For example, "200" would represent Windows Installer 2.0 and "405" would represent Windows Installer 4.5. For 64-bit Windows Installer packages, this property is set

		to 200 by default as Windows Installer 2.0 was the first version to support 64-bit packages.
InstallPrivileges	Enumeration	Use this attribute to specify the priviliges required to install the package on Windows Vista and above. This attribute's value must be one of the following: Iimited Set this value to declare that the package does not require elevated privileges to install.
		elevated Set this value to declare that the package requires elevated privileges to install. This is the default value.
InstallScope	Enumeration	Use this attribute to specify the installation scope of

this package: permachine or peruser. This attribute's value must be one of the following: perMachine Set this value to declare that the package is a per-machine installation and requires elevated privileges to install. Sets the **ALLUSERS** property to 1.

perUser

Set this value to declare that the package is a per-user installation and does not require elevated privileges to install. Sets the package's InstallPrivileges attribute to "limited."

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.
Platform	Enumeration	The platform supported by the package. This attribute's value must be one of the following: x86 Set this value to declare that the package is an x86 package. ia64 Set this value to declare that the package is an ia64 package is an ia64 package. This value requires that the InstallerVersion property be set to 200 or greater.
		Set this value to declare that the package is an x64 package. This value requires

that the InstallerVersion property be set to 200 or greater.

intel

This value has been deprecated. Use "x86" instead.

intel64

This value has been deprecated. Use "ia64" instead.

Platforms

String

This attribute has been deprecated; please use the Platform attribute instead.

ReadOnly

YesNoDefaultType The value of this

attribute conveys
whether the
package should be
opened as readonly. A database
editing tool should
not modify a readonly enforced
database and
should issue a
warning at attempts
to modify a readonly recommended

		database.
ShortNames	YesNoType	Set to 'yes' to have short filenames in the source.
SummaryCodepage	String	The code page integer value or web name for summary info strings only. See remarks for more information.

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

See Also

Wix Schema

PackageCertificates Element

Description

Digital signatures that identify installation packages in a multiproduct transaction.

Windows Installer references

MsiPackageCertificate Table

Parents

Fragment, Product

Inner Text

None

Children

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

• <u>DigitalCertificate</u> (min: 1, max: unbounded)

Attributes

None

See Also

Wix Schema

PackageGroup Element

Description

Describes a package group to a bootstrapper.

Windows Installer references

None

Parents

Fragment

Inner Text

None

Children

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

- ExePackage (min: 0, max: unbounded)
- MsiPackage (min: 0, max: unbounded)
- MspPackage (min: 0, max: unbounded)
- MsuPackage (min: 0, max: unbounded)
- PackageGroupRef (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for package group.	Yes

See Also

Wix Schema

PackageGroupRef Element

Description

Create a reference to PackageGroup element that exists inside a Bundle or Fragment element.

Windows Installer references

None

Parents

Chain, PackageGroup

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The identifier of the PackageGroup element to reference.	Yes
After	String	The identifier of a package that this group should be installed after.	

See Also

Wix Schema, PackageGroup

Patch Element

Description

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

Windows Installer references

None

Parents

Wix

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - LayoutDirectory (min: 0, max: unbounded)
 - LayoutDirectoryRef (min: 0, max: unbounded)
 - Media (min: 1, max: unbounded)
 - OptimizeCustomActions (min: 0, max: 1): Indicates whether custom actions can be skipped when applying the patch.
 - PatchFamily (min: 1, max: unbounded)
 - PatchFamilyRef (min: 0, max: unbounded)
 - <u>PatchInformation</u> (min: 0, max: 1): Optional element that allows overriding summary information properties.
 - PatchProperty (min: 0, max: unbounded)
 - <u>TargetProductCodes</u> (min: 0, max: unbounded)
 - Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - LayoutDirectory

• <u>LayoutDirectoryRef</u>

Name	Туре	Description
AllowRemoval	YesNoType	Whether thi an uninstallable patch.
ApiPatchingSymbolNoFailuresFlag	YesNoType	Flag used when creati a binary file patch. Defa is "no". Don fail patch du to imagehlp failures.
ApiPatchingSymbolNoImagehlpFlag	YesNoType	Flag used when creati a binary file patch. Defa is "no". Don use imagehlp.dl
ApiPatchingSymbolUndecoratedTooFlag	YesNoType	Flag used when creati a binary file patch. Defa is "no". Afte matching decorated symbols, try match remaining b undecorated names.
Classification	String	Category of

		updates. Recommenvalues are Critical Update, Hote Security Rollup, Security Update, Service Pac Update, Update, Update, Update Roll
ClientPatchId	String	An easily referenced identity uniq to a patch th can be used product authoring. S remarks for more information.
Codepage	String	The code painteger valu or web nam for the resulting MS See remark for more information.
Comments	String	Optional comments f browsing.
Description	String	Description the patch.
DisplayName	String	A title for the

		patch that is suitable for public displa In Add/Remov Programs fr XP SP2 on.
Id	AutogenGuid	Patch code this patch.
Manufacturer	String	Vendor releasing th package
MinorUpdateTargetRTM	YesNoType	Indicates the the patch targets the RTM version the product the most recent majo upgrade parameters and author this optional property in minor updat patches that contain sequencing information indicate that the patch removes all patches up the RTM version of the product, or to the most recent majo

		upgrade pat This propert is available beginning w Windows Installer 3.1
MoreInfoURL	String	A URL that provides information specific to the patch. In Add/Remov Programs fr XP SP2 on.
OptimizedInstallMode	YesNoType	If this attribution is set to 'yes all the patch to be applie a transaction the application of the patch optimized if possible. Available beginning with Windows Installer 3.1
OptimizePatchSizeForLargeFiles	YesNoType	When this attribute is s patches for files greater than approximate 4 MB in size may be mad smaller.

Name of the application target produsuite.

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

The ClientPatchId attribute allows you to specify an easily referenced identity that you can use in product authoring. This identity prefixes properties added by WiX to a patch transform, such as *ClientPatchId*.PatchCode and *ClientPatchId*.AllowRemoval. If the patch code GUID is auto-generated you could not reference any properties using this auto-generated prefix.

For example, if you were planning to ship a patch referred to as "QFE1" and needed to write your own registry values for Add/Remove Programs in product authoring such as the UninstallString for this patch, you could author a RegistryValue with the name UninstallString and the value [SystemFolder]msiexec.exe /package [ProductCode] /uninstall [QFE1.PatchCode]. In your patch authoring you would then set ClientPatchId to "QFE1" and WiX will add the QFE1.PatchCode property to the patch transform when the patch is created. If the Id attribute specified the patch code to be generated automatically, you could not reference the *prefix*.PatchCode property as shown above.

The summary information is automatically populated from attribute values of the Patch element including the code page. If you want to override some of these summary information properties or use a different code page for the summary information itself, author the PatchInformation element.

See Also

Wix Schema

PatchBaseline Element

Description

Identifies a set of product versions.

Windows Installer references

None

Parents

Media

Inner Text

None

Children

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

• Validate (min: 0, max: 1)

Attributes

Name	Туре	Description	Required
Id	String	Identifier for a set of product versions.	Yes

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)

• <u>DigitalCertificate</u> (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 pcp 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)
 - LayoutDirectory (min: 0, max: unbounded)
 - <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
 - PatchProperty (min: 0, max: unbounded)
 - PatchSequence (min: 0, max: unbounded)
 - ReplacePatch (min: 0, max: unbounded)
 - <u>TargetProductCode</u> (min: 0, max: unbounded)

Name	Туре	Description
Id	Guid	PatchCreation identifier primary key for identifyi

AllowMajorVersionMismatches	YesNoType	Use this to set whether versions between the u target images match. S AllowProductVersionMater for more information.
AllowProductCodeMismatches	YesNoType	Use this to set whether code between the upgraimages match. See AllowProductCodeMism more information.
CleanWorkingFolder	YesNoType	Use this to set whether should clean the temp f finished. See DontRemoveTempFold for more information.
Codepage	String	The code page integer name for the resulting F remarks for more inforn
OutputPath	String	The full path, including patch package file that generated. See PatchC more information.
SourceList	String	Used to locate the .msr patch if the cached cop See <u>PatchSourceList</u> for information.
SymbolFlags	Int	An 8-digit hex integer recombination of patch sy flags to use when creat patch. See ApiPatching for more information.
WholeFilesOnly	YesNoType	Use this to set whether should be included in the IncludeWholeFilesOnly information.

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

See Also

Wix Schema

PatchFamily Element

Description

Collection of items that should be kept from the differences between two products.

Windows Installer references

None

Parents

Fragment, Patch

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - BinaryRef (min: 0, max: unbounded)
 - ComponentRef (min: 0, max: unbounded)
 - CustomActionRef (min: 0, max: unbounded)
 - DirectoryRef (min: 0, max: unbounded)
 - FeatureRef (min: 0, max: unbounded)
 - lconRef (min: 0, max: unbounded)
 - PropertyRef (min: 0, max: unbounded)
 - <u>UIRef</u> (min: 0, max: unbounded)
 - Any Element (namespace='##other' processContents='Lax')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description	Required
Id	String	Identifier which indicates a sequence family to which	Yes

		this patch belongs.	
ProductCode	Guid	Specifies the ProductCode of the product that this family applies to.	
Supersede	YesNoType	Set this value to 'yes' to indicate that this patch will supersede all previous patches in this patch family. The default value is 'no'.	
Version	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.	Yes

See Also

Wix Schema

PatchFamilyRef Element

Description

This will cause the entire contents of the Fragment containing the referenced PatchFamily to be used in the process of creating a patch.

Windows Installer references

None

Parents

Patch

Inner Text

None

Children

None

Attributes

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

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema, PatchFamily

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

PatchFiles Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Patch, PatchCreation

Inner Text

None

Children

None

Name	Туре	Description
AdminImage	<u>YesNoType</u>	This attribute has been deprecated.
Comments	String	General purpose of the pat package. For example, "Th patch contains the logic an required to install <pre>cproduct</pre>
Compressed	<u>YesNoType</u>	This attribute has been deprecated.
Description	String	A short description of the p that includes the name of the product.
Keywords	String	A semicolon-delimited list c

		alternate sources of the pa The default is "Installer,Patching,PCP,Da
Languages	String	This attribute has been deprecated.
Manufacturer	String	The name of the manufacti the patch package.
Platforms	String	This attribute has been deprecated.
ReadOnly	YesNoDefaultType	The value of this attribute c whether the package shoul opened as read-only. A dat editing tool should not mod read-only enforced database should issue a warning at a to modify a read-only recommended database.
ShortNames	<u>YesNoType</u>	This attribute has been deprecated.
SummaryCodepage	String	The code page integer values web name for summary inf strings only. The default is See remarks for more infor

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

See Also

Wix Schema

PatchMetadata Element

Description

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

Windows Installer references

MsiPatchMetadata Table

Parents

PatchCreation

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - <u>CustomProperty</u> (min: 0, max: unbounded): A custom property that extends the standard set.
 - OptimizeCustomActions (min: 0, max: 1): Indicates whether custom actions can be skipped when applying the patch.

Name	Туре	Description	Required
AllowRemoval	YesNoType	Whether this is an uninstallable patch.	Yes
Classification	String	Category of updates. Recommended values are Critical Update, Hotfix, Security Rollup, Security Update, Service	Yes

		Pack, Update, Update Rollup.	
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.	Yes
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.	Yes

Wix Schema

PatchProperty Element

Description

A property for this patch database.

Windows Installer references

MsiPatchMetadata Table

Parents

Patch, PatchCreation

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Company	String	Name of the company for a custom metadata property.	
Name	String	Name of the patch property.	Yes
Value	String	Value of the patch property.	Yes

Remarks

When authored under the Patch element, the PatchProperty defines entries in the MsiPatchMetadata table.

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

MsiPatchSequence Table

Parents

PatchCreation

Inner Text

None

Children

None

Name	Туре	Description	Required
PatchFamily	String	Identifier which indicates a sequence family to which this patch belongs.	Yes
ProductCode	Guid	Specifies the ProductCode of the product that this family applies to. This attribute cannot the specified if the TargetImage attribute is specified.	
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	YesNoType	Set this value to 'yes' to indicate that this patch will supersede all previous patches in this patch family. The default value is 'no'.	
Target	String	This attribute has been deprecated; please use the TargetImage attribute instead.	
TargetImage	String	Specifies the TargetImage that this family applies to. This attribute cannot the specified if the ProductCode attribute is specified.	

See Also

Wix Schema

Payload Element

Description

Describes a payload to a bootstrapper.

Windows Installer references

None

Parents

<u>Container, ExePackage, MsiPackage, MspPackage, MsuPackage, PayloadGroup, UX</u>

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
DownloadUrl	String	The URL to use to download the package.	
Name	String	The destination path and file name for this payload. The default is the source file name.	
SourceFile	String	Location of the source file.	Yes

See Also

Wix Schema

PayloadGroup Element

Description

Describes a payload group to a bootstrapper.

Windows Installer references

None

Parents

Fragment

Inner Text

None

Children

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

- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for payload group.	Yes

See Also

Wix Schema

PayloadGroupRef Element

Description

Create a reference to PayloadGroup element that exists inside a Bundle or Fragment element.

Windows Installer references

None

Parents

ExePackage, MsiPackage, MspPackage, MsuPackage, PayloadGroup, UX

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The identifier of the PayloadGroup element to reference.	Yes

See Also

Wix Schema, PayloadGroup

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, Registry, RegistryKey, RegistryValue

Inner Text

None

Children

None

Name	Туре	Description
Append	<u>YesNoType</u>	
ChangePermission	<u>YesNoType</u>	
CreateChild	YesNoType	For a directory, the right to cresubdirectory. Only valid under 'CreateFolder' parent.
CreateFile	<u>YesNoType</u>	For a directory, the right to crefile in the directory. Only valid under a 'CreateFolder' parent
CreateLink	YesNoType	
CreateSubkeys	YesNoType	
Delete	<u>YesNoType</u>	

DeleteChild	YesNoType	For a directory, the right to de directory and all the files it contains, including read-only Only valid under a 'CreateFoliparent.
Domain	String	
EnumerateSubkeys	<u>YesNoType</u>	
Execute	<u>YesNoType</u>	
GenericAll	YesNoType	
GenericExecute	<u>YesNoType</u>	
GenericRead	YesNoType	specifying this will fail to grant access
GenericWrite	YesNoType	
Notify	YesNoType	
Read	YesNoType	
ReadAttributes	YesNoType	
ReadExtendedAttributes	<u>YesNoType</u>	
ReadPermission	<u>YesNoType</u>	
Synchronize	YesNoType	
TakeOwnership	YesNoType	
Traverse	YesNoType	For a directory, the right to tra the directory. By default, users assigned the BYPASS_TRAVERSE_CHEC privilege, which ignores the FILE_TRAVERSE access right Only valid under a 'CreateFoliparent.
User	String	

Write	YesNoType	
WriteAttributes	YesNoType	
WriteExtendedAttributes	<u>YesNoType</u>	

See Also

Wix Schema

PermissionEx 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 is only available when installing with MSI 5.0. For downlevel support, see the PermissionEx element from the WixUtilExtension.

Windows Installer references

MsiLockPermissionsEx Table

Parents

<u>CreateFolder</u>, <u>File</u>, <u>Registry</u>, <u>RegistryKey</u>, <u>RegistryValue</u>, ServiceInstall

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. <u>Condition</u> (min: 0, max: 1): Optional condition that controls whether the permissions are applied.

Attributes

Name	Туре	Description	Required
Id	String	Primary key used to identify this particular entry. If this is not specified the parent element's Id attribute will be used instead.	
Sddl	String	Security descriptor to apply to parent object.	Yes

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

ProcessComponents Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

<u>Wix</u>

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. <u>Package</u> (min: 1, max: 1)
- 2. Choice of elements (min: 0, max: unbounded)
 - Appld (min: 0, max: unbounded)
 - Binary (min: 0, max: unbounded)
 - ComplianceCheck (min: 0, max: unbounded)
 - Component (min: 0, max: unbounded)
 - ComponentGroup (min: 0, max: unbounded)
 - Condition (min: 0, max: unbounded)
 - CustomAction (min: 0, max: unbounded)
 - <u>CustomActionRef</u> (min: 0, max: unbounded)
 - <u>CustomTable</u> (min: 0, max: unbounded)
 - <u>Directory</u> (min: 0, max: unbounded)
 - <u>DirectoryRef</u> (min: 0, max: unbounded)
 - EmbeddedChainer (min: 0, max: unbounded)
 - EmbeddedChainerRef (min: 0, max: unbounded)

- EnsureTable (min: 0, max: unbounded)
- Feature (min: 0, max: unbounded)
- FeatureGroupRef (min: 0, max: unbounded)
- FeatureRef (min: 0, max: unbounded)
- lcon (min: 0, max: unbounded)
- InstanceTransforms (min: 0, max: unbounded)
- <u>LayoutDirectory</u> (min: 0, max: unbounded)
- <u>LayoutDirectoryRef</u> (min: 0, max: unbounded)
- MajorUpgrade (min: 0, max: unbounded)
- Media (min: 0, max: unbounded)
- <u>PackageCertificates</u> (min: 0, max: unbounded)
- PatchCertificates (min: 0, max: unbounded)
- Property (min: 0, max: unbounded)
- PropertyRef (min: 0, max: unbounded)
- SetDirectory (min: 0, max: unbounded)
- <u>SetProperty</u> (min: 0, max: unbounded)
- <u>SFPCatalog</u> (min: 0, max: unbounded)
- <u>SymbolPath</u> (min: 0, max: unbounded)
- UI (min: 0, max: unbounded)
- **UIRef** (min: 0, max: unbounded)
- Upgrade (min: 0, max: unbounded)
- WixVariable (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 - 1. <u>InstallExecuteSequence</u> (min: 0, max: 1)
 - 2. <u>InstallUISequence</u> (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')
 Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - CloseApplication

- ComPlusApplication
- ComPlusApplicationRole
- ComPlusPartition
- ComPlusPartitionRole
- Group
- HelpCollectionRef
- HelpFilter
- LayoutDirectory
- <u>LayoutDirectoryRef</u>
- MajorUpgrade
- SqlDatabase
- <u>UnitTestRef</u>
- User
- WebApplication
- WebAppPool
- WebDirProperties
- WebLog
- WebSite

Name	Туре	Description	Required
Id	AutogenGuid	The product code GUID for the product.	Yes
Codepage	String	The code page integer value or web name for the resulting MSI. See remarks for more information.	
Language	LocalizableInteger	The decimal language ID (LCID) for the product.	Yes

Manufacturer	String	The manufacturer of the product.	Yes
Name	String	The descriptive name of the product.	Yes
UpgradeCode	Guid	The upgrade code GUID for the product.	
Version	String	The product's version string.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

Remarks

You can specify any valid Windows code by integer like 1252, or by web name like Windows-1252. See <u>Code Pages</u> for more information.

See Also

Wix Schema

Progld Element

Description

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

Windows Installer references

Progld Table, Class Table, Registry Table, Icon Table

Parents

Class, Component, Progld

Inner Text

None

Children

Sequence (min: 1, max: 1)

- Progld (min: 0, max: unbounded): The version-independent Progld must be the first child element of actual Progld. Nesting other Progld elements within the Version-independent Progld will create COM+ aliases, see http://support.microsoft.com/kb/305745 for more information.
- 2. <u>Extension</u> (min: 0, max: unbounded): Extensions that refer to this Progld

Name	Туре	Description	Required
Id	String		Yes
Advertise	<u>YesNoType</u>		
Description	String		
Icon	String	For an advertised Progld, the Id of an Icon element. For a non-advertised Progld, this is the Id of a file containing an	

icon resource.

IconIndex	Integer	
NoOpen	String	Specifies that the associated Progld 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 Also

Wix Schema

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	Туре	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, Module, ODBCDataSource, ODBCDriver, Product, UI, Upgrade

Inner Text (xs:string)

This element may have inner text.

Children

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

- Sequence (min: 1, max: 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. <u>DirectorySearchRef</u> (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.

Name	Туре	Description	Re
Id	String	Unique identifier for Property.	Yes

Admin	YesNoType	Denotes that the Property is saved during admininistrative 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.
SuppressModularization	YesNoType	Use to suppress modularization of this property identifier in merge modules. Using this functionality is strongly discouraged; it should only be necessary as a

		workaround of last resort in rare scenarios.
Value	String	Sets a default value for the property. The value will be overwritten if the Property is used for a search.

Any Attribute (namespace='##other' processContents='lax') Extensib point in the WiX XML Schema. Schema extensions can register addi attributes at this point in the schema.

How Tos and Examples

• How To: Check the version number of a file during installation

See Also

Wix Schema, PropertyRef

PropertyRef Element

Description

Reference to a Property value.

Windows Installer references

None

Parents

Fragment, Module, PatchFamily, Product, UI

Inner Text

None

Children

None

Attributes

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

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

How Tos and Examples

• How To: Check for .NET Framework versions

See Also

Wix Schema, Property

ProtectFile Element

Description

Specifies a file to be protected.

Windows Installer references

None

Parents

Family

Inner Text

None

Children

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

• ProtectRange (min: 1, 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	Туре	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, UI

Inner Text (xs:string)

The element value is the optional Condition expression.

Children

None

Attributes

Name	Туре	Description	Required
Control	String	The parent Control for this Publish element, should only be specified when this element is a child of the UI element.	
Dialog	String	The parent Dialog for this Publish element, should only be specified when this element is a child of the UI element. This attribute will create a reference to the specified Dialog, so an additional DialogRef is not necessary.	
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	

sa	m	Δ	tı	m	Δ
.วต		—			C

Order	String	This attribute should only need to be set if this element is nested under a UI element in order to control the order in which this publish event will be started. If this element is nested under a Control element, the default value will be one greater than any previous Publish element's order (the first element's default value is 1). If this element is nested under a UI element, the default value is always 1 (it does not get a default value based on any previous Publish elements).
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. If the event doesn't take an attribute, a common value to use is "0".

See Also

Wix Schema

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

PublishComponents Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

PublishFeatures Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

PublishProduct Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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	Туре	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
Υ	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	Туре	Description	Required
Property	String	Property tied to this group.	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

RegisterClassInfo Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterComPlus Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterExtensionInfo Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

RegisterFonts Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterMIMEInfo Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterProduct Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterProgldInfo Element

Description

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

Windows Installer references

RegisterProgldInfo Action

Parents

AdvertiseExecuteSequence, InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterTypeLibraries Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RegisterUser Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

Wix Schema

Registry Element

Description

This element has been deprecated; please use the <u>RegistryValue</u> element instead.

Windows Installer references

Registry Table

Parents

Component, Registry

Inner Text

None

Children

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

- Permission (min: 0, max: unbounded)
- <u>PermissionEx</u> (min: 0, max: unbounded): Can also configure the ACLs for this registry key.
- Registry (min: 0, max: unbounded)
- RegistryValue (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - PermissionEx

Attributes

Name	Туре	Description
Action	Enumeration	This is the action that will be taken for this registry key. This attribute's value must be one of the following: append Appends the specified value(s) to
		a multiString registry key.

createKey

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	RegistryRootType	The predefined root key for the registry value.
Type	Enumeration	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 must be one of the following: string The value is interpreted and stored as a string (REG_SZ).
		integer The value is interpreted and stored as an integer (REG_DWORD).
		binary

stored as a hexadecimal value (REG_BINARY).

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

RegistryKey Element

Description

Used for organization of child RegistryValue elements or to create a registry key (and optionally remove it during uninstallation).

Windows Installer references

Registry Table

Parents

Component, RegistryKey

Inner Text

None

Children

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

- Permission (min: 0, max: unbounded): ACL permission
- <u>PermissionEx</u> (min: 0, max: unbounded): Can also configure the ACLs for this registry key.
- RegistryKey (min: 0, max: unbounded)
- RegistryValue (min: 0, max: unbounded)
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - PermissionEx

Attributes

Description	Required
This is the action that will be taken for this registry value. This attribute's value must be one of the following: create Creates the key if	
	This is the action that will be taken for this registry value. This attribute's value must be one of the following:

absent, when the parent component is installed.

createAndRemoveOnUninstall
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.

none

Does nothing; this element is used merely in WiX authoring for organization and does nothing to the final output. This is the default value.

ld	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. If the parent element is a RegistryKey, this value may be omitted to use the path of the parent, or if its specified it will be appended to the path of the parent.	
Root	RegistryRootType	The predefined root key for the registry value.	

How Tos and Examples

- How To: Read a registry entry during installation
- How To: Write a registry entry during installation

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)

• <u>DirectorySearch</u> (min: 0, max: 1)

• <u>DirectorySearchRef</u> (min: 0, max: 1)

• FileSearch (min: 0, max: 1)

• FileSearchRef (min: 0, max: 1)

Attributes

Name	Туре	Description	Required
Id	String	Signature to be used for the file, directory or registry key being searched for.	Yes
Key	String	Key for the registry value.	Yes
Name	String	Registry value name. If this value is null, then the value from the key's unnamed or default value, if any, is retrieved.	
Root	Enumeration	Root key for the registry value. This attribute's value must be	Yes

one of the following:

HKCR

HKEY CLASSES ROOT

HKCU

HKEY CURRENT USER

HKLM

HKEY LOCAL MACHINE

HKU

HKEY USERS

Type

Enumeration The value must be 'file' if the child is a FileSearch element, and must be 'directory' if child is a DirectorySearch element. This attribute's value must be one of the following:

directory

The registry value contains the path to a directory.

file

The registry value contains the path to a file. To return the full file path you must add a FileSearch element as a child of this element; otherwise, the parent directory of the file path is returned.

raw

Sets the raw value from the registry value. Please note that this value will contain a prefix as follows:

DWORD

Starts with '#' optionally

Yes

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 $'[\sim]'$ and ends with $'[\sim]'$.

REG SZ

No prefix, but if the first character of the registry value is '#', the installer escapes the character by prefixing it with another '#'.

Win64 YesNoType

Instructs the search to look in the 64-bit registry when the value is 'yes'. When the value is 'no', the search looks in the 32-bit registry. The default value depends on the value of the Package/@Platform attribute: if the @Platform attribute value is 'x86', the default @Win64 attribute value is 'no'; otherwise, the default value is 'yes'.

Remarks

When the Type attribute value is 'directory' the registry value must

specify the path to a directory excluding the file name. When the Type attribute value is 'file' the registry value must specify the path to a file including the file name; however, if there is no child FileSearch element the parent directory of the file is returned. The FileSearch element requires that you author the name of the file you are searching for. If you do not know the file name you must set the Type attribute to 'raw' to return the full file path including the file name.

How Tos and Examples

• How To: Read a registry entry during installation

See Also

Wix Schema, ComponentSearch, IniFileSearch

RegistrySearchRef Element

Description

References an existing RegistrySearch element.

Windows Installer references

None

Parents

Property

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Specify the Id of the RegistrySearch to reference.	Yes

See Also

Wix Schema, RegistrySearch

RegistryValue Element

Description

Used to create a registry value. For multi-string values, this can be used to prepend or append values.

For legacy authoring: 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

Component, Registry, RegistryKey

Inner Text (xs:string)

This element may have inner text.

Children

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

- MultiStringValue (min: 0, max: unbounded)
- Permission (min: 0, max: unbounded)
- <u>PermissionEx</u> (min: 0, max: unbounded): Can also configure the ACLs for this registry value.
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - <u>PermissionEx</u>

Attributes

Name	Туре	Description	Required
Action	Enumeration	This is the action that will	

be taken for this registry value. This attribute's value must be one of the following: append

Appends the specified value(s) to a multiString registry value.

prepend

Prepends the specified value(s) to a multiString registry value.

write

Writes a registry value. This is the default 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. If the parent element is a RegistryKey, this value may be omitted to use the path of the parent, or if its specified it will be appended to the path of the path of the parent.
KeyPath	<u>YesNoType</u>	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	RegistryRootType	The predefined root key for the registry value.
Type	Enumeration	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 must be one of the following: string The value is

interpreted and stored as a string (REG_SZ).

integer

The value is interpreted and stored as an integer (REG_DWORD).

binary

The value is interpreted and stored as a hexadecimal value (REG_BINARY).

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.

How Tos and Examples

• How To: Write a registry entry during installation

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

RemoveDuplicateFiles Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RemoveEnvironmentStrings Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RemoveExistingProducts Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
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

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

Inner Text

None

Children

None

Name	Туре	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	<u>WildCardLongFileNameType</u>	This attribute has been deprecated; please use the Name attribute instead.
Name	WildCardLongFileNameType	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. In prior versions of the WiX toolset, this attribute specified the short file name. This attribute's value may now be either a short or long file name. If a short file name is specified, the ShortName attribute may not be specified. If a long file name is specified. If a long file name is specified. Also, if this value is a long file name, the

ShortName attribute may be omitted to allow WiX to attempt to generate a unique short file name. However, if you wish to manually specify the short file name, then the ShortName attribute may be specified.

On <u>InstallUninstallType</u>

This value determines the time at which the file(s) may be removed. For 'install', the file will be removed only when the parent component is being installed (msiInstallStateLocal or msiInstallStateSource); for 'uninstall', the file will be removed only when the parent component is being removed (msiInstallStateAbsent); for 'both', the file will be removed in both cases.

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.

ShortName <u>WildCardShortFileNameType</u> The short file name of

The short file name of the file in 8.3 format. This attribute should only be set if you want to manually specify the short file name.

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

RemoveFiles Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

Inner Text

None

Children

None

Name	Туре	Description	Required
ld	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	InstallUninstallType	This value determines the time at which the folder may be removed, based on the install/uninstall of the parent component. For 'install', the folder will be removed only when the parent component is being installed (msilnstallStateLocal or msilnstallStateSource); for 'uninstall', the folder will be removed only when the parent component is being removed (msilnstallStateAbsent); for 'both', the folder will be removed in both cases.	Yes
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

RemoveFolders Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

Wix Schema

RemovelniValues 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

RemoveIniValues Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RemoveODBC Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RemoveRegistryKey Element

Description

Used for removing registry keys and all child keys either during install or uninstall.

Windows Installer references

Registry Table, RemoveRegistry Table

Parents

Component

Inner Text

None

Children

None

Name	Туре	Description	Required
Action	Enumeration	This is the action that will be taken for this registry value. This attribute's value must be one of the following: removeOnInstall Removes a key with all its values and subkeys when the parent component is installed.	
		removeOnUninstall Removes a key with all its values and subkeys when the parent component is uninstalled.	

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.	
String	The localizable key for the registry value.	
RegistryRootType	The predefined root key for the registry value.	
	String	this particular entry. If this attribute is not specified, an identifier will be generated by hashing the parent Component identifier, Root, Key, and Name. String The localizable key for the registry value. RegistryRootType The predefined root key for

See Also

Wix Schema

RemoveRegistryValue Element

Description

Used to remove a registry value during installation. There is no standard way to remove a single registry value during uninstall (but you can remove an entire key with RemoveRegistryKey).

Windows Installer references

RemoveRegistry Table

Parents

Component

Inner Text

None

Children

None

Name	Туре	Description	Required
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. If the parent element is a RegistryKey, this value may be omitted to use the path of the parent, or if its specified it will be appended to the path of the parent.	

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	RegistryRootType	The predefined root key for the registry value.

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

RemoveRegistryValues Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

RemoveShortcuts Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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	Туре	Description	Required
Id	Guid	Patch GUID to be unregistered if it exists on the machine targeted by this patch.	Yes

See Also

Wix Schema

RequiredPrivilege Element

Description

Privilege required by service configured by ServiceConfig parent. Valid values are a <u>privilege constant</u> or a Formatted property that resolves to a privilege constant.

Windows Installer references

MsiServiceConfig Table

Parents

ServiceConfig

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

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	A primary key that uniquely identifies this ReserveCost entry.	Yes
Directory	String	Adds the amount of disk space specified in RunFromSource or RunLocal to the volume cost of the device containing the directory. If this attribute is not set, it will default to the 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	Yes
		space to reserve if the	
		component is installed to run	
		locally.	

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

ResolveSource Action

Parents

<u>AdminExecuteSequence</u>, <u>InstallExecuteSequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the	

		position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema

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

RMCCPSearch Action

Parents

<u>InstallExecuteSequence</u>, <u>InstallUISequence</u>

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of	

		this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	If yes, this action will not occur.	

See Also

Wix Schema, CCPSearch, ComplianceCheck

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

ScheduleReboot Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Name	Туре	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.	
Overridable	YesNoType	If "yes", the sequencing of this action may be overridden by sequencing elsewhere.	
Sequence	Integer	A value used to indicate the position of this action in a sequence.	

Suppress	<u>YesNoType</u>	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

SelfRegModules Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

SelfUnregModules Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

ServiceConfig Element

Description

Configures a service being installed or one that already exists. This element's functionality is available starting with MSI 5.0.

Windows Installer references

MsiServiceConfig Table

Parents

Component, ServiceInstall

Inner Text

None

Children

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

• RequiredPrivilege (min: 0, max: unbounded): List of privileges to apply to service.

Name	Туре	Description	F
DelayedAutoStart	String	This attribute specifies whether an auto-start service should delay its start until after all other auto-start services. This attribute only affects auto-start services. Allowed values are "yes", "no" or a Formatted property that resolves to "1" (for "yes") or "0" (for "no"). If this attribute is not present the setting is not configured.	
FailureActionsWhen	String	This attribute specifies when failure actions should be applied. Allowed values are	

		"failedToStop", "failedToStopOrReturnedError" or a Formatted property that resolves to "1" (for "failedToStopOrReturnedError") or "0" (for "failedToStop"). If this attribute is not present the setting is not configured.
Id	String	Unique identifier for this service configuration. This value will default to the ServiceName attribute if not specified.
OnInstall	YesNoType	Specifies whether to configure the service when the parent Component is installed. This attribute may be combined with OnReinstall and OnUninstall.
OnReinstall	YesNoType	Specifies whether to configure the service when the parent Component is reinstalled. This attribute may be combined with OnInstall and OnUninstall.
OnUninstall	YesNoType	Specifies whether to configure the service when the parent Component is uninstalled. This attribute may be combined with OnInstall and OnReinstall.
PreShutdownDelay	String	This attribute specifies time in milliseconds that the Service Control Manager (SCM) waits after notifying the service of a system shutdown. If this attribute is not present the default value, 3 minutes, is used.

ServiceName	String	Specifies the name of the service to configure. This value will default to the ServiceInstall/@Name attribute when nested under a ServiceInstall element.
ServiceSid	String	Specifies the service SID to apply to the service. Valid values are "none", "restricted", "unrestricted" or a Formatted property that resolves to "0" (for "none"), "3" (for "restricted") or "1" (for "unrestricted"). If this attribute is not present the setting is not configured.

See Also

Wix Schema

ServiceConfigFailureActions Element

Description

Configures the failure actions for a service being installed or one that already exists. This element's functionality is available starting with MSI 5.0.

Windows Installer references

MsiServiceConfigFailureActions Table

Parents

Component, ServiceInstall

Inner Text

None

Children

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

• Failure (min: 0, max: unbounded): Ordered list of failure actions to apply to service.

Name	Туре	Description	Required
Command	String	This attribute specifies command to execute when a "runCommand" failure action hit. If an empty string is provided it clears the existing command. If this attribute is not present the setting is not changed.	
Id	String	Unique identifier for this service configuration.	

		This value will default to the ServiceName attribute if not specified.
OnInstall	YesNoType	Specifies whether to configure the service when the parent Component is installed. This attribute may be combined with OnReinstall and OnUninstall.
OnReinstall	YesNoType	Specifies whether to configure the service when the parent Component is reinstalled. This attribute may be combined with OnInstall and OnUninstall.
OnUninstall	YesNoType	Specifies whether to configure the service when the parent Component is uninstalled. This attribute may be combined with OnInstall and OnReinstall.
RebootMessage	String	Specifies the message to show for a reboot failure action. If an empty string is provided it clears any existing reboot message. If this attribute is not present the setting is not changed.

ResetPeriod	String	Specifies the time in seconds to reset the failure count. If this attribute is not present the failure count will not be reset.
ServiceName	String	Specifies the name of the service to configure. This value will default to the ServiceInstall/@Name attribute when nested under a ServiceInstall element.

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. <u>ServiceArgument</u> (min: 0, max: unbounded): Ordered list of arguments used when modifying services.

Name	Туре	Description	Required
Id	String		Yes
Name	String	Name of the service.	Yes
Remove	InstallUninstallType	Specifies whether the service should be removed by the DeleteServices action on install, uninstall or both. For 'install', the service will be removed only	

when the parent component is being installed (msiInstallStateLocal or msiInstallStateSource); for 'uninstall', the service will be removed only when the parent component is being removed (msiInstallStateAbsent); for 'both', the service will be removed in both cases.

Start

<u>InstallUninstallType</u>

Specifies whether the service should be started by the StartServices action on install, uninstall or both. For 'install', the service will be started only when the parent component is being installed (msiInstallStateLocal or msiInstallStateSource); for 'uninstall', the service will be started only when the parent component is being removed (msiInstallStateAbsent); for 'both', the service will be started in both cases.

Stop

<u>InstallUninstallType</u>

Specifies whether the service should be stopped by the StopServices action on install, uninstall or both. For 'install', the service

will be stopped only
when the parent
component is being
installed
(msiInstallStateLocal or
msiInstallStateSource);
for 'uninstall', the service
will be stopped only
when the parent
component is being
removed
(msiInstallStateAbsent);
for 'both', the service will
be stopped in both
cases.

Wait <u>YesNoType</u>

Specifies whether or not to wait for the service to complete before continuing. The default is 'yes'.

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	Туре	Description	Required
Id	String	 The value of this attribute should be one of the following: The name (not the display name) of a previously installed service. A foreign key referring to another ServiceInstall/@Id. 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

ServiceInstall Element

Description

Adds and removes services for parent Component.

Windows Installer references

ServiceInstall Table

Parents

Component

Inner Text

None

Children

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

- <u>PermissionEx</u> (min: 0, max: unbounded): Configures the ACLs for this service.
- <u>ServiceConfig</u> (min: 0, max: unbounded)
- <u>ServiceConfigFailureActions</u> (min: 0, max: unbounded)
- <u>ServiceDependency</u> (min: 0, max: unbounded): ordered list of dependencies when installing services
- Any Element (namespace='##other' processContents='Lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional elements at this point in the schema.
 - ServiceConfig

Name	Туре	Description	Required
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 must be one of the following: ignore Logs the error and continues with the startup operation. normal Logs the error, displays a message box and continues the startup operation. critical	Yes

Logs the error if it
is possible and the
system is
restarted with the
last configuration
known to be good.
If the last-known-
good configuration
is being started,
the startup
operation fails.

Id	String	Unique identifier for this service configuration. This value will default to the Name attribute if not specified.	
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 does not	Yes

support boot or system. This attribute's value must be one of the following: auto

The service will start during startup of the system.

demand

The service will start when the service control manager calls the StartService function.

disabled

The service can no longer be started.

boot

The service is a device driver that will be started by the operating system boot loader. This value is not currently supported by the Windows Installer.

system

The service is a device driver that will be started by the IolnitSystem function. This value is not

currently supported by the Windows Installer.

Type Enumeration The Windows Installer Yes does not currently support kernelDriver or systemDriver This attribute's value must be one of the following: *ownProcess* A Win32 service that runs its own process. shareProcess A Win32 service that shares a process. kernelDriver A kernel driver service. This value is not currently supported by the Windows Installer. systemDriver A file system driver service. This value is not currently supported by the Windows Installer. 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

SetDirectory Element

Description

Sets a Directory to a particular value. This is accomplished by creating a Type 51 custom action that is appropriately scheduled in the InstallUISequence and InstallExecuteSequence.

Windows Installer references

CustomAction Table

Parents

Fragment, Module, Product

Inner Text (xs:string)

The condition that determines whether the Directory is set. If the condition evaluates to false, the SetDirectory is skipped.

Children

None

Name	Туре	Description	Required
Id	String	This attribute specifies a reference to a Directory element with matching Id attribute. The path of the Directory will be set to the Value attribute.	
Sequence	SequenceType	Controls which sequences the Directory assignment is sequenced in. For 'execute', the assignment is scheduled in the InstallExecuteSequence. For 'ui', the assignment is scheduled in the	

		InstallUISequence. For 'both', the assignment is scheduled in both the InstallUISequence and the InstallExecuteSequence. The default is both.	
Value	String	This attribute specifies a string value to assign to the Directory. The value can be a literal value or derived from a Property element using the Formatted syntax.	

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

<u>Wix Schema</u>, <u>Custom</u>, <u>CustomActionRef</u>, <u>InstallUISequence</u>, <u>InstallExecuteSequence</u>

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

SetODBCFolders Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

Wix Schema

SetProperty Element

Description

Sets a Property to a particular value. This is accomplished by creating a Type 51 custom action that is appropriately scheduled in the InstallUISequence and InstallExecuteSequence.

Windows Installer references

CustomAction Table

Parents

Fragment, Module, Product

Inner Text (xs:string)

The condition that determines whether the Property is set. If the condition evaluates to false, the Set is skipped.

Children

None

Name	Туре	Description	Required
After	String	The name of the standard or custom action after which this action should be performed. Mutually exclusive with the Before attribute. A Before or After attribute is required when setting a Property.	
Before	String	The name of the standard or custom action before which this action should be performed. Mutually exclusive with the After attribute. A Before or After	

		attribute is required when setting a Property.
Id	String	This attribute specifies the Property to set to the Value.
Sequence	SequenceType	Controls which sequences the Property assignment is sequenced in. For 'execute', the assignment is scheduled in the InstallExecuteSequence. For 'ui', the assignment is scheduled in the InstallUISequence. For 'both', the assignment is scheduled in both the InstallUISequence and the InstallUISequence and the InstallExecuteSequence. The default is both.
Value	String	This attribute specifies a string value to assign to the Property. The value can be a literal value or derived from a Property element using the Formatted syntax.

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

<u>Wix Schema, Custom, CustomActionRef, InstallUISequence, InstallExecuteSequence</u>

SFPCatalog Element

Description

Adds a system file protection update catalog file

Windows Installer references

SFPCatalog Table

Parents

Fragment, Module, Product, SFPCatalog

Inner Text

None

Children

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

- SFPCatalog (min: 0, max: unbounded)
- SFPFile (min: 0, max: unbounded): Primary Key to File Table.

Attributes

Name	Туре	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	Туре	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

Inner Text

None

Children

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

• <u>Icon</u> (min: 0, max: unbounded)

• ShortcutProperty (min: 0, max: unbounded)

Name	Туре	Description
Id	String	Unique identifier for th shortcut. This value w as the primary key for
Advertise	YesNoType	Specifies if the shortco be advertised or not. It advertised shortcuts a point at a particular ap- identified by a Product and should not be sha- between applications. Advertised shortcuts of for the most recently in application, and are re-

		when that application removed. The default 'no'.
Arguments	String	The command-line are for the shortcut. Note resolution of propertie Arguments field is limi property formatted as in this field can only be if the property already intended value when to component owning the is installed. For example argument "[#MyDoc.d resolve to the correct same process must be the file MyDoc.doc and component that owns shortcut.
Description	String	The localizable descri the shortcut.
DescriptionResourceDII	String	The Formatted string the full path to the land neutral file containing Manifest. Generally at using [#filekey] form. Vattribute is specified, to DescriptionResourcel must also be provided
		This attribute is only u Windows Vista and at this attribute is not speand the install is runni Vista and above, the vame attribute is uthis attribute is provide

		install is running on Vi above, the value in the attribute is ignored.
DescriptionResourceId	Integer	The description name the shortcut. This mus non-negative number. this attribute is specific DescriptionResource attribute must also be populated.
		This attribute is only u Windows Vista and at this attribute is not speand the install is runni Vista and above, the value is attribute is popula the install is running o and above, the value i Name attribute is igno
Directory	String	Identifier reference to element where shortco created. When nested Component element, that attribute's value will detect the parent directory. Contact this attribute is require
DisplayResourceDII	String	The Formatted string the full path to the lang neutral file containing Manifest. Generally at using [#filekey] form. \attribute is specified, t DisplayResourceId att must also be provided

		This attribute is only u Windows Vista and at this attribute is not populated and the install is runni Vista and above, the value is attribute is populated the install is running of and above, the value install is running of and above, the value is ignored.
DisplayResourceId	Integer	The display name indeshortcut. This must be negative number. Whe attribute is specified, to DisplayResourceDII at must also be provided
		This attribute is only u Windows Vista and at this attribute is not spe and the install is runni Vista and above, the vithe Name attribute is uthis attribute is specificate install is running of and above, the value in Name attribute is ignored
Hotkey	Integer	The hotkey for the she low-order byte contain virtual-key code for the the high-order byte comodifier flags. This minon-negative number, of installation package generally recommend this option, because the add duplicate hotkeys users desktop. In additional additional contains the shock of t

		practice of assigning hashortcuts can be prob- users using hotkeys for accessibility.
Icon	String	Identifier reference to element. The Icon identifier should have the same extension as the file the points at. For example shortcut to an executa "my.exe") should refer Icon with identifier like "Mylcon.exe"
IconIndex	Integer	Identifier reference to element.
LongName	LongFileNameType	This attribute has bee deprecated; please us Name attribute instead
Name	LongFileNameType	In prior versions of the toolset, this attribute s the short name. This a value may now be eith or long name. If a sho specified, the ShortNa attribute may not be s a long name is specifical LongName attribute m specified. Also, if this long name, the ShortNattribute may be omittallow WiX to attempt t generate a unique short However, if this name with another shortcut wish to manually spec short name, then the

		ShortName attribute n specified.
ShortName	<u>ShortFileNameType</u>	The short name of the in 8.3 format. This attr should only be set if the conflict between gene short names or the us to manually specify the name.
Show	Enumeration	This attribute's value r one of the following: normal The shortcut targed displayed using the SW_SHOWNORI attribute.
		minimized The shortcut targed displayed using the SW_SHOWMINN attribute.
		maximized The shortcut targed displayed using the SW_SHOWMAXI attribute.
Target	String	This attribute can only this Shortcut element under a Component e When nested under a Component element, attribute's value will de the parent directory. Tattribute's value is the a non-advertised shor

		attribute is not valid fo advertised shortcuts. I specify this value, its v should be a property i enclosed by square bi]), that is expanded in or a folder pointed to I shortcut.
WorkingDirectory	String	Directory identifier (or identifier that resolves directory) that resolve path of the working directory.

How Tos and Examples

• How To: Create a shortcut on the Start Menu

See Also

Wix Schema

ShortcutProperty Element

Description

Property values for a shortcut. This element's functionality is available starting with MSI 5.0.

Windows Installer references

MsiShortcutProperty Table

Parents

Shortcut

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Unique identifier for MsiShortcutProperty table. If omitted, a stable identifier will be generated from the parent shortcut identifier and Key value.	
Key	String	A formatted string identifying the property to be set.	Yes
Value	String	A formatted string supplying the value of the property.	

See Also

Wix Schema, Shortcut

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	Туре	Description	Required
After	String		
Before	String		
Dialog	String		Yes
OnExit	ExitType	mutually exclusive with Before, After, and Sequence attributes	
Overridable	YesNoType	If "yes", the sequencing of this dialog may be overridden by sequencing elsewhere.	
Sequence	Integer		

See Also

Wix Schema

StartServices Element

Description

Starts system services. The condition for this action may be specified in the element's inner text.

Windows Installer references

StartServices Action

Parents

<u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

StopServices Action

Parents

<u>InstallExecuteSequence</u>

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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	Туре	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	Туре	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

<u>Component, Directory, ExternalFile, File, Media, Product, TargetFile, TargetImage, UpgradeFile, UpgradeImage</u>

Inner Text

None

Children

None

Attributes

Name	Туре	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	Туре	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

Upgradelmage

Inner Text

None

Children

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

• SymbolPath (min: 0, max: unbounded)

• TargetFile (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	String	Identifier for the target image.	Yes
IgnoreMissingFiles	<u>YesNoType</u>	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

TargetProductCode Element

Description

A product code for a product that can accept the patch.

Windows Installer references

None

Parents

PatchCreation, TargetProductCodes

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The product code for a product that can accept the patch. This can be '*'. See remarks for more information.	Yes

Remarks

When using the PatchCreation element, if the Id attribute value is '*' or this element is not authored, the product codes of all products referenced by the TargetImages element are used.

When using the Patch element, the Id attribute value must not be '*'. Use the TargetProductCodes/@Replace attribute instead.

See Also

Wix Schema

TargetProductCodes Element

Description

The product codes for products that can accept the patch.

Windows Installer references

None

Parents

Patch

Inner Text

None

Children

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

• <u>TargetProductCode</u> (min: 1, max: unbounded)

Attributes

Name	Туре	Description	Required
Replace	YesNoType	Whether to replace the product codes that can accept the patch from the target packages with the child elements.	

See Also

Wix Schema

Text Element

Description

An alternative to using the Text attribute when the value contains special XML characters like <, >, or &.

Windows Installer references

None

Parents

Control

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

<u>TextStyle Table</u>

Parents

<u>UI</u>

Inner Text

None

Children

None

Attributes

Туре	Description	Required
String		Yes
Integer	0 to 255	
<u>YesNoType</u>		
String		Yes
Integer	0 to 255	
<u>YesNoType</u>		
Integer	0 to 255	
String		Yes
YesNoType		
<u>YesNoType</u>		
	String Integer YesNoType String Integer YesNoType Integer String YesNoType YesNoType	String Integer 0 to 255 YesNoType String Integer 0 to 255 YesNoType Integer 0 to 255 YesNoType String YesNoType Integer 0 to 255

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

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	Туре	Description	Required
Id	Guid	The GUID that identifes 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. The default value is 'no'.	
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	Integer	The major version of the type library. The value should be an integer from 0 - 255.
MinorVersion	Integer	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

UI Element

Description

Enclosing element to compartmentalize UI specifications.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

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

- <u>BillboardAction</u> (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
- <u>Dialog</u> (min: 0, max: unbounded): Dialog specification, called from Sequence
- <u>DialogRef</u> (min: 0, max: unbounded): Reference to a Dialog specification.
- <u>EmbeddedUI</u> (min: 0, max: unbounded): Embedded UI definition with EmbeddedResource children.
- Error (min: 0, max: unbounded): Error text associated with install error
- <u>ListBox</u> (min: 0, max: unbounded): ListBox table with ListItem children
- <u>ListView</u> (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)
- PropertyRef (min: 0, max: unbounded)
- Publish (min: 0, max: unbounded)
- RadioButtonGroup (min: 0, max: unbounded): RadioButton table with RadioButton children
- <u>TextStyle</u> (min: 0, max: unbounded): TextStyle entry for use in control text
- <u>UIRef</u> (min: 0, max: unbounded)
- <u>UIText</u> (min: 0, max: unbounded): values for UIText property, not installer Property
- Sequence (min: 1, max: 1)
 - 1. AdminUISequence (min: 0, max: 1)
 - 2. <u>InstallUISequence</u> (min: 0, max: 1)

Attributes

Name	Туре	Description	Required
Id	String		

See Also

Wix Schema, UIRef

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, Module, PatchFamily, Product, UI

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String		Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

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	Туре	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

UnpublishComponents Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnpublishFeatures Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnregisterClassInfo Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnregisterComPlus Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

<u>UnregisterExtensionInfo Action</u>

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnregisterFonts Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnregisterMIMEInfo Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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 Progld information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

UnregisterProgIdInfo Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

UnregisterTypeLibraries Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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, Product

Inner Text

None

Children

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

- Property (min: 0, max: unbounded): Nesting a Property element under an Upgrade element has been deprecated. Please nest Property elements in any of the other supported locations.
- <u>UpgradeVersion</u> (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	<u>Guid</u>	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

<u>UpgradeImage</u>

Inner Text

None

Children

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

• SymbolPath (min: 0, max: unbounded)

Attributes

Name	Туре	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

Upgradelmage 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)
 - <u>UpgradeFile</u> (min: 0, max: unbounded)

Attributes

Name	Туре	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

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	Туре	Description	Requ
ExcludeLanguages	<u>YesNoType</u>	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 "no" to make the range of versions detected exclude 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. Windows Installer documentation for the Upgrade table states that the property specified in this field must be a public property and must be added to the SecureCustomProperties property. WiX automatically appends the property specified in this field to the SecureCustomProperties property when creating an MSI. Each UpgradeVersion must have a unique Property value. After the FindRelatedProducts action is run, the value of this property is a list product codes, separated by semicolons (;), detected on the system.	Yes
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 commadelimited 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') Extensit point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Wix Schema

UX Element

Description

Contains all the relevant information about the setup UI.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

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

- Payload (min: 0, max: unbounded)
- PayloadGroupRef (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Name	String	The destination path and file name for this UX payload. The default is the source file name. Use this attribute to rename the UX entry point or extract it into a subfolder.	
SourceFile	String	The entry point of the UX.	

See Also

Wix Schema

Validate Element

Description

Sets information in the patch transform that determines if the transform applies to an installed product and what errors should be ignored when applying the patch transform.

Windows Installer references

None

Parents

PatchBaseline

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Rec
IgnoreAddExistingRow	YesNoType	Ignore errors when adding existing rows. The default is 'yes'.	
IgnoreAddExistingTable	<u>YesNoType</u>	Ignore errors when adding existing tables. The default is 'yes'.	
IgnoreChangingCodePage	YesNoType	Ignore errors when changing the database code page. The default is 'no'.	
IgnoreDeleteMissingRow	<u>YesNoType</u>	Ignore errors when	

		deleting missing rows. The default is 'yes'.
IgnoreDeleteMissingTable	<u>YesNoType</u>	Ignore errors when deleting missing tables. The default is 'yes'.
IgnoreUpdateMissingRow	<u>YesNoType</u>	Ignore errors when updating missing rows. The default is 'yes'.
ProductId	YesNoType	Requires that the installed ProductCode match the target ProductCode used to create the transform. The default is 'yes'.
ProductLanguage	YesNoType	Requires that the installed ProductLanguage match the target ProductLanguage used to create the transform. The default is 'no'.
ProductVersion	Enumeration	Determines how many fields of the installed ProductVersion to compare. See remarks for more information. The default is 'Update'. This attribute's

value must be one of the following: Major Checks the major version.

Minor

Checks the major and minor versions.

Update

Checks the major, minor, and update versions.

ProductVersionOperator

Enumeration Determines how the

installed ProductVersion is compared to the target **ProductVersion** used to create the transform. See remarks for more information. The default is 'Equal'. This attribute's value must be one of the following: Lesser

Installed ProductVersion < target ProductVersion.

LesserOrEqual Installed **ProductVersion**

<= target ProductVersion.

Equal

Installed

ProductVersion

= target

ProductVersion.

GreaterOrEqual

Installed

ProductVersion

>= target

ProductVersion.

Greater

Installed

ProductVersion

> target

ProductVersion.

UpgradeCode

<u>YesNoType</u>

Requires that the

installed

UpgradeCode match the target

UpgradeCode used to create the

transform. The default is 'yes'.

Remarks

A transform contains the differences between the target product and the upgraded product. When a transform or a patch (which contains transforms) is applied, the following properties of the installed product are validated against the properties of the target product stored in a transform.

- ProductCode
- ProductLanguage
- ProductVersion

• UpgradeCode

Windows Installer simply validates that the ProductCode, ProductLanguage, and UpgradeCode of an installed product are equivalent to those propeties of the target product used to create the transform; however, the ProductVersion can be validated with a greater range of comparisons.

You can compare up to the first three fields of the ProductVersion. Changes to the fourth field are not validated and are useful for small updates. You can also choose how to compare the target ProductVersion used to create the transform with the installed ProductVersion. For example, while the default value of 'Equals' is recommended, if you wanted a minor upgrade patch to apply to the target ProductVersion and all older products with the same ProductCode, you would use 'LesserOrEqual'.

See Also

Wix Schema

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

ValidateProductID Action

Parents

InstallExecuteSequence, InstallUISequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

Suppress <u>YesNoType</u> If yes, this action will not occur.

See Also

Wix Schema

Variable Element

Description

Describes a burn engine variable to define.

Windows Installer references

None

Parents

Bundle

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Name	String	The name for the variable.	Yes
Type	Enumeration	Type of the variable, inferred from the value if not specified. This attribute's value must be one of the following: string numeric version	
Value	String	Starting value for the variable.	Yes

See Also

Verb Element

Description

Verb definition for an Extension. When advertised, this element creates a row in the <u>Verb table</u>. When not advertised, this element creates the appropriate rows in <u>Registry table</u>.

Windows Installer references

Verb Table, Registry Table

Parents

Extension

Inner Text

None

Children

None

Attributes

Name	Туре	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	This attribute has been deprecated; please use the TargetFile attribute instead.	
TargetFile	String	Either this attribute or the TargetProperty attribute must be specified for a non-advertised verb. The value should be the identifier of the target file to be executed for the verb.	
TargetProperty	String	Either this attribute or the TargetFile attribute must be specified for a non-advertised verb. The value should be the identifier of the property which will resolve to the path to the target file to be executed for the verb.	

See Also

Wix Element

Description

This is the top-level container element for every wxs file. Among the possible children, the Bundle, Product, Module, Patch, 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 pcp 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)
 - Bundle (min: 0, max: 1)
 - Module (min: 0, max: 1)
 - Patch (min: 0, max: 1)
 - Product (min: 0, max: 1)
 - 2. Fragment (min: 0, max: unbounded)

Attributes

RequiredVersion VersionType	Required version of the WiX toolset this input file.
	tother' processContents='lax') Extensit sions can register additional attributes a
RequiredVersion String	The version of this extension require compile the defining source. (http://schemas.microsoft.com/wix/P

See Also

Wix Schema

WixVariable Element

Description

This element exposes advanced WiX functionality. Use this element to declare WiX variables from directly within your authoring. WiX variables are not resolved until the final msi/msm/pcp file is actually generated. WiX variables do not persist into the msi/msm/pcp file, so they cannot be used when an MSI file is being installed; it's a WiX-only concept.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	The name of the variable.	Yes
Overridable	YesNoType	Set this value to 'yes' in order to make the variable's value overridable either by another WixVariable entry or via the command-line option - d <name>=<value> for light.exe. If the same variable is declared overridable in multiple places it will cause an error (since WiX won't know which value is correct).</value></name>	

		The default value is 'no'.
Value	String	The value of the variable. The Yes value cannot be an empty string because that would make it possible to accidentally set a column to null.

See Also

Wix Schema

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

WriteEnvironmentStrings Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

WriteIniValues Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	<u>YesNoType</u>	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

WriteRegistryValues Action

Parents

InstallExecuteSequence

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	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

AutogenGuid (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 setting the value to "*". Also allows "PUT-GUID-HERE" for use in examples.

Pattern Type

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'.$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Wix Schema

Wixloc 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 Xml Localization files (.wxl).

Root Element

WixLocalization

Target Namespace

http://schemas.microsoft.com/wix/2006/localization

Document Should Look Like

<?xml version="1.0"?> <<u>WixLocalization</u> xmlns="http://schemas.microsoft.com/wix/2006/localization"></WixLocalization>

String Element

Description

None

Windows Installer references

None

Parents

WixLocalization

Inner Text (xs:string)

This element may have inner text.

Children

None

Name	Туре	Description	Required
Id	String	Identity of the resource.	Yes
Localizable	LocalizationYesNoType	Indicates whether the string is localizable text or a non-localizable string that must be unique per locale. No WiX tools are affected by the value of this attribute; it used as documentation for localizers to ignore things like GUIDs or	

		identifiers that look like text.
Overridable	LocalizationYesNoType	Determines if the localized string may be overridden.

How Tos and Examples

- How To: Build a localized version of your installer
- How To: Make your installer localizable

See Also

Wixloc Schema

WixLocalization Element

Description

None

Windows Installer references

None

Parents

None

Inner Text

None

Children

Sequence (min: 0, max: unbounded)

1. String (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Codepage	String	The code page integer value or web name for the resulting database. You can also specify -1 which will not reset the database code page. See remarks for more information.	
Culture	String	Culture of the localization strings.	Yes

Remarks

You can specify any valid Windows code by integer like 1252, or by web name like Windows-1252 or iso-8859-1. See <u>Code Pages</u> for more information.

How Tos and Examples

• How To: Build a localized version of your installer

• How To: Make your installer localizable

See Also

Wixloc Schema

LocalizationYesNoType (Simple Type)

Description

None

Enumeration Type

Possible values: {no, yes}

See Also

Wixloc Schema

Complus 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 COM+ Extension.

Target Namespace

http://schemas.microsoft.com/wix/ComPlusExtension

Child Elements

- ComPlusApplication
- ComPlusApplicationRole
- ComPlusAssembly
- <u>ComPlusAssemblyDependency</u>
- ComPlusComponent
- ComPlusGroupInApplicationRole
- ComPlusGroupInPartitionRole
- ComPlusInterface
- ComPlusMethod
- ComPlusPartition
- ComPlusPartitionRole
- ComPlusPartitionUser
- ComPlusRoleForComponent
- ComPlusRoleForInterface

- ComPlusRoleForMethod
- ComPlusSubscription
- <u>ComPlusUserInApplicationRole</u>
- <u>ComPlusUserInPartitionRole</u>

Complus Extension)

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 locater, 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, Component, Fragment, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - <u>ComPlusApplicationRole</u> (min: 0, max: unbounded)
 - ComPlusAssembly (min: 0, max: unbounded)

Name	Туре	Description
Id	String	Identifier for the ele
AccessChecksLevel	Enumeration	This attribute's value one of the follow

		applicationLevel
Activation	Enumeration	This attribute's value one of the following in proc
		local
ApplicationAccessChecksEnabled	<u>YesNoType</u>	
ApplicationDirectory	String	
ApplicationId	String	Id for the application attribute can be on which case an id with generated on instate element is a locate attribute can be on value is provided for Name attribute.
Authentication	Enumeration	This attribute's value one of the follow default
		none
		connect
		call
		packet
		integrity
		privacy
AuthenticationCapability	Enumeration	This attribute's value one of the follownone
		secureReference

staticCloaking dynamicCloaking

Name	String	Name of the applic
MaxDumpCount	Int	
IsEnabled	YesNoType	
		impersonate delegate
		identify
ImpersonationLevel	Enumeration	This attribute's value one of the followanonymous
Identity	String	
EventsEnabled	<u>YesNoType</u>	
DumpPath	String	
DumpOnFailfast	<u>YesNoType</u>	
DumpOnException	<u>YesNoType</u>	
DumpEnabled	<u>YesNoType</u>	
Description	String	
Deleteable	<u>YesNoType</u>	
CRMLogFile	String	
CRMEnabled	YesNoType	
CreatedBy	String	
ConcurrentApps	Int	
CommandLine	String	
Changeable	<u>YesNoType</u>	

		This attribute can I omitted if the elem locater, and a valu provided for the Pa attribute.
Partition	String	If the element is not of a ComPlusPartitelement, this attribute be provided with the ComPlusPartition or representing the puthe application below.
Password	String	
QCAuthenticateMsgs	Enumeration	This attribute's value one of the follow secureApps
		on
QCListenerMaxThreads	Int	
QueueListenerEnabled	YesNoType	
QueuingEnabled	YesNoType	
RecycleActivationLimit	Int	
RecycleCallLimit	Int	
RecycleExpirationTimeout	Int	
RecycleLifetimeLimit	Int	
RecycleMemoryLimit	Int	
Replicable	<u>YesNoType</u>	
Replicable RunForever	YesNoType YesNoType	

SoapActivated	<u>YesNoType</u>	
SoapBaseUrl	String	
SoapMailTo	String	
SoapVRoot	String	
SRPEnabled	<u>YesNoType</u>	
SRPTrustLevel	Enumeration	This attribute's value one of the following disallowed
		fullyTrusted
ThreeGigSupportEnabled	YesNoType	

Complus Schema

Complus ApplicationRole Element (Complus Extension)

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 locater, referencing an existing application role.

Windows Installer references

None

Parents

ComPlusApplication, Component, Fragment, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - <u>ComPlusGroupInApplicationRole</u> (min: 0, max: unbounded)
 - ComPlusUserInApplicationRole (min: 0, max: unbounded)

Name	Туре	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	

Complus Schema

Complus Extension)

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, Component

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)

Name	Туре	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 DIIPathFromGAC is set to "yes".
DIIPath	String	The path to locate the assembly DLL during registration. This attribute should be provided if DIIPathFromGAC is not set to "yes".
DIIPathFromGAC	YesNoType	Indicates that the DLL path should be extracted from the GAC instead for being provided in the DIIPath 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.
PSDIIPath	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 must be one of the following: native .net	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 there 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

Complus Schema

ComPlusAssemblyDependency Element (Complus Extension)

Description

Defines a dependency between two assemblies. This element affects the order in which assembles are registered. Any assemblies referenced by this element are guarantied 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	Туре	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

Complus Schema

Complus Component Element (Complus Extension)

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)
 - <u>ComPlusRoleForComponent</u> (min: 0, max: unbounded)
 - ComPlusSubscription (min: 0, max: unbounded)

Name	Туре	Description
Id	String	Identifier for telement.
AllowInprocSubscribers	<u>YesNoType</u>	
CLSID	<u>Uuid</u>	CLSID of the component.
ComponentAccessChecksEnabled	<u>YesNoType</u>	
ComponentTransactionTimeout	Int	
Component Transaction Time out Enabled	<u>YesNoType</u>	

COMTIIntrinsics	<u>YesNoType</u>	
ConstructionEnabled	<u>YesNoType</u>	
ConstructorString	String	
CreationTimeout	Int	
Description	String	
EventTrackingEnabled	<u>YesNoType</u>	
ExceptionClass	String	
FireInParallel	<u>YesNoType</u>	
IISIntrinsics	<u>YesNoType</u>	
InitializesServerApplication	<u>YesNoType</u>	
IsEnabled	<u>YesNoType</u>	
IsPrivateComponent	<u>YesNoType</u>	
JustInTimeActivation	YesNoType	
LoadBalancingSupported	YesNoType	
MaxPoolSize	Int	
MinPoolSize	Int	
MultiInterfacePublisherFilterCLSID	String	
MustRunInClientContext	YesNoType	
MustRunInDefaultContext	YesNoType	
ObjectPoolingEnabled	<u>YesNoType</u>	
PublisherID	String	
SoapAssemblyName	String	
SoapTypeName	String	
Synchronization	Enumeration	This attribute value must b

		one of the following: ignored
		none
		supported
		required
		requiresNew
Transaction	Enumeration	This attribute value must b one of the following: ignored
		none
		supported
		required
		requiresNew
TxIsolationLevel	Enumeration	This attribute value must b one of the following: any
		readUnComr
		readCommitt
		repeatableRe
		serializable

Complus Schema

ComPlusGroupInApplicationRole Element (Complus Extension)

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 locater element. To reference a locater element use the ApplicationRole attribute.

Windows Installer references

None

Parents

ComPlusApplicationRole, Component

Inner Text

None

Children

None

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	Yes	
		table.		

Complus Schema

ComPlusGroupInPartitionRole Element (Complus Extension)

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, Component

Inner Text

None

Children

None

Attributes

Name	Туре	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

Complus Schema

ComPlusInterface Element (Complus Extension)

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)
 - <u>ComPlusRoleForInterface</u> (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	String	Identifier for the element.	Yes
Description	String		
IID	<u>Uuid</u>	IID of the interface.	Yes
QueuingEnabled	YesNoType		

See Also

Complus Schema

ComPlusMethod Element (Complus Extension)

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)

Name	Туре	Description	Required
Id	String	Identifier for the element.	Yes
AutoComplete	<u>YesNoType</u>		
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.	

Complus Schema

ComPlusPartition Element (Complus Extension)

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 locater, referencing an existing partition.

Windows Installer references

None

Parents

Component, Fragment, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Choice of elements (min: 0, max: unbounded)
 - <u>ComPlusApplication</u> (min: 0, max: unbounded)
 - ComPlusPartitionRole (min: 0, max: unbounded)
 - ComPlusPartitionUser (min: 0, max: unbounded)

Name	Туре	Description	Required
Id	String	Identifier for the element.	Yes
Changeable	<u>YesNoType</u>		
Deleteable	YesNoType		
Description	String		

Name	String	Name of the partition. This attribute can be omitted if the element is a locater, 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 locater, this attribute can be omitted if a value is provided for the Name attribute.

Complus Schema

Complus PartitionRole Element (Complus Extension)

Description

Defines a COM+ partition role. Partition roles can not be created; this element can only be used as a locater to reference an existing role.

Windows Installer references

None

Parents

ComPlusPartition, Component, Fragment, Module, Product

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	Туре	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

Complus Schema

Complus PartitionUser Element (Complus Extension)

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, Component

Inner Text

None

Children

None

Attributes

Name	Туре	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

Complus Schema

ComPlusRoleForComponent Element (Complus Extension)

Description

Represents a role assignment to a COM+ component.

Windows Installer references

None

Parents

ComPlusComponent, Component

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

Complus Schema

Complus RoleForInterface Element (Complus Extension)

Description

Represents a role assignment to an interface.

Windows Installer references

None

Parents

ComPlusInterface, Component

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

Complus Schema

Complus RoleForMethod Element (Complus Extension)

Description

Represents a role assignment to a COM+ method.

Windows Installer references

None

Parents

ComPlusMethod, Component

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

Complus Schema

Complus Subscription Element (Complus Extension)

Description

Defines an event subscription for a COM+ component.

Windows Installer references

None

Parents

ComPlusComponent, Component

Inner Text

None

Children

None

Attributes

Name	Туре	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

Complus Schema

ComPlusUserInApplicationRole Element (Complus Extension)

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 locater element. To reference a locater element use the ApplicationRole attribute.

Windows Installer references

None

Parents

ComPlusApplicationRole, Component

Inner Text

None

Children

None

Attributes

Name	Туре	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	Yes
		table.	

See Also

Complus Schema

ComPlusUserInPartitionRole Element (Complus Extension)

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, Component

Inner Text

None

Children

None

Attributes

Name	Туре	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

Complus Schema

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'$.

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

uuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF".

Pattern Type

Must match the regular expression: $[0-9A-Fa-f]{8}\-?[0-9A-Fa-f]{4}\-?[0-9A-Fa-f]{4}\-?[0-9A-Fa-f]{12}'$.

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z.]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Difxapp 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 Driver Install Frameworks for Applications Extension.

Target Namespace

http://schemas.microsoft.com/wix/DifxAppExtension

Child Elements

Driver

Driver Element (Difxapp Extension)

Description

Installs a driver. To use this element, you need to reference the WixDifxAppExtension extension and add the .wixlib appropriate for the target platform (difxapp_x86.wixlib, difxapp_x64.wixlib, or difxapp_ia64.wixlib) to your project.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	Description
AddRemovePrograms	YesNoType	Specifies that the DIFxApp Cust entry in the Add/Remove Prograthe default is 'yes'.
DeleteFiles	YesNoType	If set to "yes", configures DIFxA that were copied to the system 1 a driver package was installed. "no" or not present, DIFxApp do from a system. Note that configures files is controlled by the F component that represents the component that repre

Specifies that the DIFxApp Cust
the installation of a new Plug an even if the currently installed dri better match than the new drive excellent way to ensure the DIF recognize the Component conta The default is null which means install a driver via DIFxApp Cushttp://www.microsoft.com/whdc/for more information.
If set to "yes", configures DIFxA driver packages and driver pack For more information, see "Insta Packages in Legacy Mode" earl attribute is set to "no" or not pre only signed driver packages. No DIFxApp to install unsigned driver package in the MsiDriver driver package in the MsiDriver Setting DriverLegacy to "yes" see the Flags entry value. Setting Designed the bit in the Flags entry value the bit in the Flags entry value the present, DIFxApp uses a defaul
Specifies that the DIFxApp Cust the user to connect the Plug and connected. The default is 'yes'.
Specifies an optional installatior DIFxApp CustomActions install installation package in the order numbers. The same sequence r more than one driver; however, packages with the same sequer installed cannot be determined.

Difxapp Schema

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

Difxapp 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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

Difxapp Schema

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'$.

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]=([.\%])=([.\%])$ [_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Firewall 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 Firewall Extension.

Target Namespace

http://schemas.microsoft.com/wix/FirewallExtension

Child Elements

- FirewallException
- RemoteAddress

Version 3.5.1623.0

FirewallException Element (Firewall Extension)

Description

Registers an exception for a program or a specific port and protocol in the Windows Firewall on Windows XP SP2, Windows Server 2003 SP1, and later. For more information about the Windows Firewall, see About Windows Firewall API.

Windows Installer references

None

Parents

Component, File

Inner Text

None

Children

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

• RemoteAddress (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
ld	String	Unique ID of this firewall exception.	Yes
File	String	Identifier of a file to be granted access to all incoming ports and protocols. If you use File, you cannot also use Program, Port, or Protocol.	
IgnoreFailure	YesNoType	If "yes," failures to register this firewall exception will	

		be silently ignored. If "no" (the default), failures will cause rollback.	
Name	String	Name of this firewall exception, visible to the user in the firewall control panel.	Yes
Port	String	Port to allow through the firewall for this exception. If you use Port, you cannot also use File or Program.	
Profile	Enumeration	Profile type for this firewall exception. Default is "all". This attribute's value must be one of the following: domain	
		private	
		public	
		pablic	
		all	
Program	String	•	

specify Port and you cannot also use File or Program. This attribute's value must be one of the following: tcp

udp

Scope

Enumeration The scope of this firewall exception, which indicates whether incoming connections can come from any computer including those on the Internet or only those on the local network subnet. To more precisely specify allowed remote address, specify a custom scope using RemoteAddress child elements. This attribute's value must be one of the following: any

localSubnet

See Also

Firewall Schema

Version 3.5.1623.0

RemoteAddress Element (Firewall Extension)

Description

A remote address to which the port or program can listen. Address formats vary based on the version of Windows and Windows Firewall the program is being installed on. For Windows XP SP2 and Windows Server 2003 SP1, see RemoteAddresses Property. For Windows Vista and Windows Server 2008, see RemoteAddresses Property.

Windows Installer references

None

Parents

FirewallException

Inner Text (xs:string)

A remote address.

Children

None

Attributes

None

See Also

Firewall Schema

Version 3.5.1623.0

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'$.

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]=([.\%])=([.\%])$ [_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Gaming 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 Gaming Extension.

Target Namespace

http://schemas.microsoft.com/wix/GamingExtension

Child Elements

- Game
- PlayTask
- SupportTask

Version 3.5.1623.0

Game Element (Gaming Extension)

Description

Registers a game in Game Explorer on Windows Vista and later. The executable must have an embedded Game Definition File. For more information about Game Explorer and GDFs, see The Windows Vista Game Explorer. This registration is accomplished via custom action.

On Windows XP, this element instead records the same information in the registry so that later upgrades to Windows Vista register the game in Game Explorer.

Windows Installer references

None

Parents

File

Inner Text

None

Children

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

- PlayTask (min: 0, max: unbounded)
- SupportTask (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	<u>Guid</u>	The game's instance ID.	Yes
ExecutableFile	String	Identifier of the file that is the game's executable, if it isn't the parent file.	
GdfResourceFile	String	Identifier of the file that contains the game's GDF	

resource, if it doesn't exist in the parent file.

See Also

Gaming Schema

PlayTask Element (Gaming Extension)

Description

Creates a shortcut to the parent File and registers it as a "play task" in Game Explorer. For more information, see <u>Game Explorer Tasks</u>. PlayTask should not be used when authoring the tasks in the GDF using ExtendedProperties\GameTasks available in Windows 7.

Windows Installer references

None

Parents

Game

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Arguments	String	Command-line arguments to be passed to the game executable for this task.	
Name	String	User-visible task name Game Explorer shows on its context menu. Note that the first task is named "Play" regardless of the name you provide.	Yes

See Also

SupportTask Element (Gaming Extension)

Description

Creates an Internet shortcut and registers it as a "support task" in Game Explorer. For more information, see <u>Game Explorer Tasks</u>. SupportTask should not be used when authoring the tasks in the GDF using ExtendedProperties\GameTasks available in Windows 7.

Windows Installer references

None

Parents

Game

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Address	String	URI for this task.	
Name	String	User-visible task name Game Explorer shows on its context menu. Note that the first task is named "Play" regardless of the name you provide.	Yes

See Also

IsRichSavedGame Attribute (Gaminç Extension)

Description

Registers this extension for the <u>rich saved games</u> property handler on Windows Vista and later.

Windows Installer references

None

Parents

Extension

See Also

Gaming Schema

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-HERE|([!\$])\(loc\.[_A-Za-z][0-9A-Za-z_.]*\)|!\(wix\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus. A-Za-z][0-9A-Za-z].$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z.]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

lis 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 Internet Information Services Extension.

Target Namespace

http://schemas.microsoft.com/wix/IIsExtension

Child Elements

- Certificate
- CertificateRef
- <u>HttpHeader</u>
- MimeMap
- RecycleTime
- WebAddress
- WebApplication
- WebApplicationExtension
- WebAppPool
- WebDir
- WebDirProperties
- WebError
- WebFilter
- WebLog
- WebProperty
- WebServiceExtension
- WebSite

• WebVirtualDir

Certificate Element (lis Extension)

Description

Used to install and uninstall certificates.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	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	

		atribute 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	<u>YesNoType</u>		
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 must be one of the following: currentUser	Yes

localMachine

StoreName Enumeration This attribute's value Yes must be one of the following: ca 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. 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.

trustedPeople

Contains the certificates of those directly trusted people and resources. See MSDN documentation for more information.

trustedPublisher

Contains the certificates of those publishers who are trusted. See MSDN documentation for more information.

See Also

lis Schema, CertificateRef

CertificateRef Element (lis Extension)

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

lis Schema, Certificate

HttpHeader Element (lis Extension)

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	Туре	Description	Required
Id	String	Primary key for custom HTTP Header entry. This will default to the Name attribute.	
Name	String	Name of the custom HTTP Header.	Yes
Value	String	Value for the custom HTTP Header. This attribute can contain a formatted string that is processed at install time to insert the values of properties using [PropertyName] syntax. Also supported are environment variables, file installation paths, and component installation directories; see Formatted for details.	

See Also lis Schema

MimeMap Element (lis Extension)

Description

MimeMap definition for IIS resources.

Windows Installer references

None

Parents

WebSite, WebVirtualDir

Inner Text

None

Children

None

Attributes

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

See Also

Iis Schema

RecycleTime Element (lis Extension)

Description

IIS6 Application Pool Recycle Times on 24 hour clock.

Windows Installer references

None

Parents

WebAppPool

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Value	String	Pattern: '\d{1,2}:\d{2}'.	Yes

See Also

Iis Schema

WebAddress Element (lis Extension)

Description

WebAddress for WebSite

Windows Installer references

None

Parents

WebSite

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String		Yes
Header	String		
IP	String	 The IP address to locate an existing WebSite or create a new WebSite. When the WebAddress is part of a WebSite element used to locate an existing web site the following rules are used: When this attribute is not specified only the "All Unassigned" IP address will be located. When this attribute is explicitly specified only the specified IP address will be located. 	

 When this attribute has the value "*" then any IP address including the "All Unassigned" IP address will be located

When the WebAddress is part of a WebSite element used to create a new web site the following rules are used:

- When this attribute is not specified or the value is "*" the "All Unassigned" IP address will be used.
- When this attribute is explicitly specified the IP address will use that value.

The IP attribute can contain a formatted string that is processed at install time to insert the values of properties using [PropertyName] syntax.

Port	String		Yes
Secure	YesNoType	Determines if this address represents a secure binding. The default is 'no'.	

See Also

Iis Schema

WebApplication Element (lis Extension)

Description

Defines properties for a web application. These properties can be used for more than one application defined in a web site or vroot, by defining this element in a common location and referring to it by setting the WebApplication attribute of the WebSite and WebVirtualDir elements.

Windows Installer references

None

Parents

Fragment, Module, Product, WebSite, WebVirtualDir

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. WebApplicationExtension (min: 0, max: unbounded)

Name	Туре	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 must be one of the following: VBScript JScript
Isolation	Enumeration	Sets the application isolation level for this application for pre-IIS 6 applications. This attribute's value must be one of the following: Neans the application

exe	ecutes
with	nin the
IIS	process.

medium

Executes pooled in a separate process.

high

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

<u>lis Schema</u>

WebApplicationExtension Element (lis Extension)

Description

Extension for WebApplication

Windows Installer references

None

Parents

WebApplication

Inner Text

None

Children

None

Name	Туре	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	<u>YesNoType</u>	
Verbs	String	

See Also

<u>lis Schema</u>

WebAppPool Element (lis Extension

Description

IIS6 Application Pool

Windows Installer references

None

Parents

Component, Fragment, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. RecycleTime (min: 0, max: unbounded)

Name	Туре	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 must be one of the following: none	
		shutdown	

Identity	Enumeration	Identity you want the AppPool to run under. Use the 'other' value in conjunction with the User attribute to specify nonstandard user. This attribute's value must 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

<u>lis Schema</u>

WebDir Element (lis Extension)

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, WebSite

Inner Text

None

Children

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

• WebDirProperties (min: 0, max: 1)

Name	Туре	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. This attribute may not be specified if a WebDirProperties element is directly nested in this element.	
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

<u>lis Schema</u>

WebDirProperties Element (lis Extension)

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, Module, Product, WebDir, WebSite, WebVirtualDir

Inner Text

None

Children

None

Name	Туре	Description
Id	String	
AccessSSL	YesNoType	A value of true indic that file access requ SSL file permission processing, with or without a client certificate. This corresponds to AccessSSL flag for AccessSSLFlags IIS

		metabase property.
AccessSL128	YesNoType	A value of true indic that file access requ SSL file permission processing with a minimum key size o bits, with or without client certificate. Thi corresponds to AccessSSL128 flag AccessSSLFlags IIS metabase property.
AccessSSLMapCert	YesNoType	This corresponds to AccessSSLMapCer for AccessSSLFlags metabase property.
AccessSSLNegotiateCert	<u>YesNoType</u>	This corresponds to AccessSSLNegotiat flag for AccessSSLF IIS metabase prope
AccessSSLRequireCert	<u>YesNoType</u>	This corresponds to AccessSSLRequired flag for AccessSSLF IIS metabase prope
AnonymousAccess	YesNoType	Sets the Enable Anonymous Access checkbox, which may anonymous users to Windows user accommendated Windows user accommendated When setting this to you should also protente user account us the AnonymousUse attribute, and determed what setting to use the

		IIsControlledPasswo attribute. Defaults to
AnonymousUser	String	Reference to the Id attribute on the Use element to be used the anonymous use the directory. See th User element for moinformation.
AspDetailedError	YesNoType	Sets the option for whether to send det ASP errors back to 1 client on script error Default is 'no.'
AuthenticationProviders	String	Comma delimited lis order of precedence Windows authentica providers that IIS wi attempt to use: NTL Kerberos, Negotiate others.
BasicAuthentication	YesNoType	Sets the Basic Authentication optio which allows clients provide credentials i plaintext over the wi Defaults to 'no.'
CacheControlCustom	String	Custom HTTP 1.1 c control directives.
CacheControlMaxAge	NonNegativeInteger	Integer value specify the cache control maximum age value
ClearCustomError	<u>YesNoType</u>	Specifies whether II return custom errors this directory.

DefaultDocuments	String	The list of default documents to set fo web directory, in cor delimited format.
DigestAuthentication	YesNoType	Sets the Digest Authentication optio which allows using authentication with domain user accour Defaults to 'no.'
Execute	<u>YesNoType</u>	
HttpExpires	String	Value to set the HttpExpires attribute for a Web Dir in the metabase.
IIsControlledPassword	YesNoType	Sets whether IIS should be control the password used for the Window account specified in AnonymousUser attribute. Defaults to
Index	YesNoType	Sets the Index Rescoption, which specific whether this web directory should be indexed. Defaults to
LogVisits	YesNoType	Sets whether visits to site should be logge Defaults to 'no.'
PassportAuthentication	<u>YesNoType</u>	Sets the Passport Authentication optio which allows clients provide credentials .Net Passport accou

Read YesNoType Script YesNoType WindowsAuthentication YesNoType Sets the Windows Authentication optio which enables integ Windows authentica to be used on the si Defaults to 'no.' Write YesNoType

See Also

<u>Iis Schema</u>

WebError Element (lis Extension)

Description

Custom Web Errors used by WebSites and Virtual Directories.

Windows Installer references

None

Parents

WebSite, WebVirtualDir

Inner Text

None

Children

None

Attributes

Name	Туре	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 lis Schema

WebFilter Element (lis Extension)

Description

IIs Filter for a Component

Windows Installer references

None

Parents

Component, WebSite

Inner Text

None

Children

None

Name	Туре	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	The legal values are "first", "last", or a number. If a number is specified, it must be greater than 0.	
Name	String	The name of the filter to be used in IIS.	Yes
Path	String	The path of the filter executable	Yes

		file. This should usually be a value like '[!FileId]', where 'FileId' is the file identifier of the filter executable file.	
WebSite	String	Specifies the parent website for this filter (if there is one). If this is a global filter, then this attribute should not be specified.	

See Also

<u>lis Schema</u>

WebLog Element (lis Extension)

Description

WebLog definition.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

None

Name	Туре	Description	Required
Id	String	Identifier for the WebLog.	Yes
Туре	Enumeration	This attribute's value must be one of the following: IIS Microsoft IIS Log File Format	Yes
		NCSA NCSA Common Log File Format	
		none Disables logging.	
		ODBC Logging	
		W3C W3C Extended Log File	

Format

See Also

<u>lis Schema</u>

WebProperty Element (lis Extension

Description

IIS Properties

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	Enumeration	This attribute's value must be one of the following: ETagChangeNumber	Yes
		IIs5IsolationMode	
		MaxGlobalBandwidth	
		LogInUTF8	
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

lis Schema

WebServiceExtension Element (lis Extension)

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

Inner Text

None

Children

None

Name	Туре	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 lis Schema

WebSite Element (lis Extension)

Description

IIs Web Site

Windows Installer references

None

Parents

Component, Fragment, Module, Product

Inner Text

None

Children

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

- <u>CertificateRef</u> (min: 0, max: unbounded)
- <u>HttpHeader</u> (min: 0, max: unbounded)
- MimeMap (min: 0, max: unbounded)
- WebAddress (min: 1, max: unbounded)
- WebApplication (min: 0, max: 1)
- WebDir (min: 0, max: unbounded)
- WebDirProperties (min: 0, max: 1)
- WebError (min: 0, max: unbounded)
- WebFilter (min: 0, max: unbounded)
- WebVirtualDir (min: 0, max: unbounded)

Name	Туре	Description	Requi
Id	String	Identifier for the WebSite. Used within the MSI package only.	Yes
AutoStart	<u>YesNoType</u>	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	References the Id attribute for a WebDirProperties element that specifies the security and	

		access properties for this website root directory. This attribute may not be specified if a WebDirProperties element is directly nested in this element.
Sequence	Integer	Sequence that the web site is to be created in.
SiteId	String	Optional attribute to directly specify the site id of the WebSite. Use this to ensure all web sites in a web garden get the same site id. If a number is provided, the site id must be unique on all target machines. If "*" is used, the Description attribute will be hashed to create a unique value for the site id. This value must be a positive number or a "*" or a formatted value that resolves to "-1"

		(for the same behavior as "*") or a positive number or blank. If this attribute is absent then the web site will be located using the WebAddress element associated with the web site.
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 lis Schema

WebVirtualDir Element (lis Extension)

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, WebSite, WebVirtualDir

Inner Text

None

Children

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

- HttpHeader (min: 0, max: unbounded)
- MimeMap (min: 0, max: unbounded)
- WebApplication (min: 0, max: 1)
- WebDirProperties (min: 0, max: 1)
- WebError (min: 0, max: unbounded)
- WebVirtualDir (min: 0, max: unbounded)

Name	Туре	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. This attribute may not be specified if a WebDirProperties element is directly nested in this element.	
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

<u>lis Schema</u>

AutogenGuid (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 setting the value to "*". Also allows "PUT-GUID-HERE" for use in examples.

Pattern Type

See Also

lis 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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

Iis Schema

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

<u>Iis Schema</u>

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

lis 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

lis Schema

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

Iis 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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'$.

See Also

lis Schema

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]=([.\%])=([.\%])$ [_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

<u>Iis Schema</u>

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

<u>Iis Schema</u>

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

Iis Schema

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

Iis Schema

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

lis 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

Iis Schema

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

lis Schema

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\{[]\] \{1,16\}(\.[^{\})<:/''+,;=\{[]\] \{0,6\})?|([!$])(loc\.[_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

Iis 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

Iis Schema

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

<u>Iis Schema</u>

IsolatedApp 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 Isolated Applications.

Root Element

IsolatedApp

Target Namespace

http://wix.sourceforge.net/schemas/clickthrough/isolatedapp/2006

Document Should Look Like

<?xml version="1.0"?> <<u>IsolatedApp</u> xmlns="http://wix.sourceforge.net/schemas/clickthrough/isolatedarce."
.

/IsolatedApp>

Application Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

IsolatedApp

Inner Text

None

Children

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

- <u>Details</u> (min: 0, max: 1)
- EntryPoint (min: 0, max: 1)
- <u>lcon</u> (min: 0, max: 1)
- <u>Id</u> (min: 0, max: 1)
- Name (min: 0, max: 1)
- <u>Source</u> (min: 0, max: 1)

Attributes

None

See Also

IsolatedApp Schema

Description Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Details Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

EntryPoint Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
PackageVersion	Boolean		

See Also

IsolatedApp Schema

Feed Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Icon Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application, Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Index	Integer		

See Also

IsolatedApp Schema

Id Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application, Package

Inner Text (uuid)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

IsolatedApp Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

None

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Package (min: 1, max: 1)
- 2. Application (min: 1, max: 1)
- 3. PreviousFeed (min: 0, max: 1)

Attributes

None

See Also

IsolatedApp Schema

Manufacturer Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Name Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Package Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

IsolatedApp

Inner Text

None

Children

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

- Description (min: 0, max: 1)
- Feed (min: 0, max: 1)
- <u>lcon</u> (min: 0, max: 1)
- <u>Id</u> (min: 0, max: 1)
- Manufacturer (min: 0, max: 1)
- UpdateRate (min: 0, max: 1)
- Version (min: 0, max: 1)

Attributes

None

See Also

IsolatedApp Schema

PreviousFeed Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

IsolatedApp

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Source Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

UpdateRate Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:integer)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp Schema

Version Element (IsolatedApp Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (VersionType)

This element may have inner text.

Children

None

Attributes

None

See Also

IsolatedApp 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

IsolatedApp 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

IsolatedApp Schema

Lux 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 Lux Extension.

Target Namespace

http://schemas.microsoft.com/wix/2009/Lux

Child Elements

- Condition
- Expression
- Mutation
- UnitTest
- UnitTestRef

Condition Element (Lux Extension)

Description

Conditions for a unit test.

Windows Installer references

None

Parents

UnitTest

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

Lux Schema

Expression Element (Lux Extension)

Description

An expression that must evaluate to true to succeed. Cannot be specified if Property is specified.

Windows Installer references

None

Parents

UnitTest

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

Lux Schema

Mutation Element (Lux Extension)

Description

Test mutations let you author unit tests with different expected results. The mutation id is passed as the value of the WIXLUX_RUNNING_MUTATION property. Your custom action, typically in an '#ifdef DEBUG' block, can retrieve the WIXLUX_RUNNING_MUTATION property and hard-code different behavior based on the mutation. To author test mutations, use the Mutation element with UnitTest elements as children.

Windows Installer references

None

Parents

Fragment

Inner Text

None

Children

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

• <u>UnitTest</u> (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Value of the WIXLUX_RUNNING_MUTATION property set by the mutation.	

See Also

Lux Schema

Version 3.5.1623.0

UnitTest Element (Lux Extension)

Description

Describes a unit test to be executed against a particular custom action, resulting in a particular property value. Single-value properties, multi-value properties, and name/value-pair properties can all be tested declaratively.

Windows Installer references

None

Parents

Fragment, Mutation, UnitTest

Inner Text

None

Children

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

- Condition (min: 0, max: unbounded)
- Expression (min: 0, max: unbounded)
- <u>UnitTest</u> (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Requ
CustomAction	String	Identifier of the custom action to be tested.	
Id	String	Identifier of the unit test. If you omit this attribute, a stable identifier is generated for you.	
Index	String	A formatted string that evaluates to either an integer index into a multi-value property or a	

		string name of a name/value-pair multi- value property.
NameValueSeparator	String	One character that is used to separate values in a name/value-pair multi-value property.
Operator	Enumeration	The operator to apply to the property and value. Default is "equal." This attribute's value must be one of the following: equal (Default) Compares Property to Value and succeeds if they are equal. notEqual Compares Property to Value and succeeds if they are
		NOT equal. caseInsensitiveEqual Compares Property to Value and succeeds if they are equal (ignoring case).
		caseInsensitiveNotEqual Compares Property to Value and succeeds if they are NOT equal (ignoring case).
Property	String	Name of the property

		set by the custom action.
Value	String	The value to compare to the property.
ValueSeparator	String	One character that is used to separate values in a multi-value property.

See Also

Lux Schema

Version 3.5.1623.0

UnitTestRef Element (Lux Extension

Description

Identifies a unit test to be linked into a project.

Windows Installer references

None

Parents

Fragment, Product

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Identifier of the unit test to be included.	

See Also

Lux Schema

Version 3.5.1623.0

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'.$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Msmq 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 MSMQ Extension.

Target Namespace

http://schemas.microsoft.com/wix/MsmqExtension

Child Elements

- MessageQueue
- MessageQueuePermission

Version 3.5.1623.0

MessageQueue Element (Msmq Extension)

Description

None

Windows Installer references

None

Parents

Component

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. MessageQueuePermission (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Required
Id	String		Yes
Authenticate	<u>YesNoType</u>	Default: No.	
BasePriority	Integer		
Journal	<u>YesNoType</u>	Default: No.	
JournalQuota	Integer		
Label	String		Yes
MulticastAddress	String		
PathName	String		Yes
PrivLevel	Enumeration	This attribute's value must be one of the	

following: none optional body

Quota	Integer		
ServiceTypeGuid	String		
Transactional	<u>YesNoType</u>	Default: No.	

See Also

Msmq Schema

Version 3.5.1623.0

MessageQueuePermission Element (Msmq Extension)

Description

None

Windows Installer references

None

Parents

Component, MessageQueue

Inner Text

None

Children

None

Attributes

Type	Description	Required
String		Yes
<u>YesNoType</u>		
String		
String		
<u>YesNoType</u>		
	String YesNoType YesNoType YesNoType YesNoType YesNoType YesNoType YesNoType String String	String YesNoType YesNoType YesNoType YesNoType YesNoType YesNoType String String String

<u>YesNoType</u>
<u>YesNoType</u>
String
<u>YesNoType</u>

See Also

Msmq Schema

Version 3.5.1623.0

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z]+\)'$.

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

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

Child Elements

Nativelmage

Nativelmage Element (Netfx Extension)

Description

Improves the performance of managed applications by creating native images. Requires the .NET Framework 2.0 or newer to be installed on the target machine since it runs NGEn.

Windows Installer references

None

Parents

File

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Re
Id	String	The identifier for this Nativelmage.	Ye
AppBaseDirectory	String	The identifier of the directory to use for locating dependent assemblies. For DLL assemblies and assemblies installed to the Global Assembly Cache (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: specifiying 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	<u>YesNoType</u>	Set to "yes" to generate native images that can be used under a debugger. The default value is "no".
Dependencies	<u>YesNoType</u>	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 must be one of the following: 32bit

Attempt to generate native images only for the 32-bit version of the .NET Framework on the target machine. If the 32-bit version of the .NET Framework 2.0 or newer is not present on the target machine, native image custom actions will not be scheduled. This is the default value.

64bit

Attempt to generate native images only for the 64-bit version of the .NET Framework on the target machine. If a 64-bit version of the .NET Framework 2.0 or newer is not present on the target machine, native image custom actions will not be scheduled.

all

Attempt to generate native images for the 32-bit and 64-bit versions of the .NET Framework on the target machine. If a version of the .NET Framework 2.0 or newer is not present on the target machine for a processor

architecture, native image custom actions will not be scheduled for that processor architecture.

Priority

Enumeration Sets the priority of generating the native images for this assembly. This attribute's value must be one of the following:

0

This is the highest priority, it means that image generation occurs syncronously during the setup process. This option will slow down setup performance.

1

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.

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	<u>YesNoType</u>	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. The native image custom actions are configured to ignore failures so that failing to generate or remove a native image will not cause setup to fail and roll back.

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

Netfx Schema

OfficeAddin 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 Office Addins.

Root Element

OfficeAddin

Target Namespace

http://wix.sourceforge.net/schemas/clickthrough/officeaddin/2006

Document Should Look Like

<?xml version="1.0"?> <<u>OfficeAddin</u> xmlns="http://wix.sourceforge.net/schemas/clickthrough/officeaddin">/OfficeAddin>

Application Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

OfficeAddin

Inner Text

None

Children

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

- Details (min: 0, max: 1)
- EntryPoint (min: 0, max: 1)
- ExtendsApplication (min: 0, max: 1)
- <u>lcon</u> (min: 0, max: 1)
- <u>Id</u> (min: 0, max: 1)
- Name (min: 0, max: 1)
- <u>Source</u> (min: 0, max: 1)

Attributes

None

See Also

OfficeAddin Schema

Description Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Details Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

EntryPoint Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

ExtendsApplication Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (SupportedOfficeApplications)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Feed Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Icon Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application, Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Туре	Description	Required
Index	Integer		

See Also

OfficeAddin Schema

Id Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application, Package

Inner Text (uuid)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Manufacturer Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Name Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

OfficeAddin Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

None

Inner Text

None

Children

Sequence (min: 1, max: 1)

- 1. Package (min: 1, max: 1)
- 2. Application (min: 1, max: 1)
- 3. PreviousFeed (min: 0, max: 1)

Attributes

None

See Also

OfficeAddin Schema

Package Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

OfficeAddin

Inner Text

None

Children

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

- Description (min: 0, max: 1)
- Feed (min: 0, max: 1)
- <u>lcon</u> (min: 0, max: 1)
- <u>Id</u> (min: 0, max: 1)
- Manufacturer (min: 0, max: 1)
- UpdateRate (min: 0, max: 1)
- Version (min: 0, max: 1)

Attributes

None

See Also

OfficeAddin Schema

PreviousFeed Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

OfficeAddin

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Source Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Application

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

UpdateRate Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (xs:integer)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

Version Element (OfficeAddin Extension)

Description

None

Windows Installer references

None

Parents

Package

Inner Text (VersionType)

This element may have inner text.

Children

None

Attributes

None

See Also

OfficeAddin Schema

SupportedOfficeApplications (Simple Type)

Description

None

Enumeration Type

Possible values: {Excel2003, Outlook2003, PowerPoint2003, Word2003}

See Also

OfficeAddin 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

OfficeAddin 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

OfficeAddin Schema

Ps 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 PowerShell Extension.

Target Namespace

http://schemas.microsoft.com/wix/PSExtension

Child Elements

- FormatsFile
- Snapln
- <u>TypesFile</u>

FormatsFile Element (Ps Extension)

Description

Identifies the parent File as a formats XML file for the referenced PowerShell snap-in.

Windows Installer references

None

Parents

File, Snapln

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
FileId	String	Reference to the formats File ID. This is required when nested under the SnapIn element.	
SnapIn	String	Reference to the PowerShell snap- in ID for which this formats file is associated. This is required when nested under the File element.	

Remarks

A formats XML file that defines output formats for objects on the pipeline.

See Also

Ps Schema

SnapIn Element (Ps Extension)

Description

Identifies the parent File as a PowerShell snap-in to be registered on the system.

Windows Installer references

None

Parents

File

Inner Text

None

Children

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

• FormatsFile (min: 0, max: unbounded)

• TypesFile (min: 0, max: unbounded)

Attributes

Name	Туре	Description	Re
Id	String	The identifier for this PowerShell snap-in.	Yes
AssemblyName	String	This attribute has been deprecated.	
CustomSnapInType	String	The full type name of a class that is used to	

		register a list of cmdlets and providers.
Description	String	A brief description of the snap-in.
DescriptionIndirect	EmbeddedResource	An embedded resource that contains a brief description of the snapin. This resource must be embedded in the current snap-in assembly.
RequiredPowerShellVersion	VersionType	The required version of PowerShell that must be installed and is associated with the snap-in registration. The default value is

		"1.0".
Vendor	String	The name of the snap-in vendor.
VendorIndirect	EmbeddedResource	An embedded resource that contains the name of the snap-in vendor. This resource must be embedded in the current snap-in assembly.
Version	VersionType	The version of the snapin. If not specified, this is taken from the assembly name.

Remarks

<u>PowerShell</u> snap-ins allow developers to extend the functionality of of the PowerShell engine. Add this element to identify the parent File as a PowerShell snap-in that will get registered on the system.

See Also

TypesFile Element (Ps Extension)

Description

Identifies the parent File as a types XML file for the referenced PowerShell snap-in.

Windows Installer references

None

Parents

File, SnapIn

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
FileId	String	Reference to the types File ID. This is required when nested under the SnapIn element.	
Snapln	String	Reference to the PowerShell snap- in ID for which this types file is associated. This is required when nested under the File element.	

Remarks

A types XML file used by the extensible type system.

See Also

RequiredVersion Attribute (Ps Extension)

Description

The version of this extension required to compile the defining source.

Windows Installer references

None

Parents

Wix

See Also

Ps Schema

Version 3.5.1623.0

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

EmbeddedResource (Simple Type)

Description

Values should be in the format ResourceName, StringName, where ResourceName is the name of the embedded resource in your assembly sans the ".resources" extension, and StringName is the name of the string resource in the embedded resource.

Example: UtilityMshSnapInResources,Description

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'.$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]=([.\%])=([.\%])$ [_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

VersionType (Simple Type)

Description

Values of this type will look like: "x", "x.x", "x.x.x", or "x.x.x.x" where x is an integer from 0 to 65534.

Pattern Type

Must match the regular expression: $\d{1,5}(\.\d{1,5}){0,3}'$.

See Also

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Sql 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 SQL Server Extension.

Target Namespace

http://schemas.microsoft.com/wix/SqlExtension

Child Elements

- SqlDatabase
- SqlFileSpec
- SqlLogFileSpec
- SqlScript
- SqlString

Version 3.5.1623.0

SqlDatabase Element (Sql Extension)

Description

SQL Database

Windows Installer references

None

Parents

Component, Fragment, Module, Product

Inner Text

None

Children

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

- SqlScript (min: 0, max: unbounded)
- SqlString (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 - 1. SqlFileSpec (min: 0, max: 1)
 - 2. SqlLogFileSpec (min: 0, max: 1)

Attributes

Name	Туре	Description	Required
Id	String		Yes
ConfirmOverwrite	<u>YesNoType</u>		
ContinueOnError	<u>YesNoType</u>		
CreateOnInstall	YesNoType		
CreateOnReinstall	YesNoType	Specifies whether to create the database when the associated	

CreateOnUninstall	<u>YesNoType</u>		
Database	String	The name of the database. The value can be a literal value or derived from a Property element using the Formatted syntax.	Yes
DropOnInstall	<u>YesNoType</u>		
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

Sql Schema, User

Version 3.5.1623.0

SqlFileSpec Element (Sql Extension)

Description

File specification for a Sql database.

Windows Installer references

None

Parents

SqlDatabase

Inner Text

None

Children

None

Attributes

Name	Туре	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 GB, MB and KB and % suffixes can be used to specify gigabytes, megabytes, kilobytes or a percentage of the current file size to grow. The default is megabytes if no suffix is specified. The default value is 10% if GrowthSize is not specified, and the minimum value is 64 KB. The GrowthSize setting for a file cannot exceed the	

MaxSize setting.

MaxSize	String	Specifies the maximum size to which the database file can grow. The GB, MB and KB suffixes can be used to to specify gigabytes, megabytes or kilobytes. The default is megabytes if no suffix is specified. If MaxSize is not specified, the file will grow until the disk is full.
Name	String	Specifies the logical name for the database file.
Size	String	Specifies the size of the database file. The GB, MB and KB suffixes can be used to specify gigabytes, megabytes or kilobytes. The default is megabytes if no suffix is specified. 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

Sql Schema

Version 3.5.1623.0

SqlLogFileSpec Element (Sql Extension)

Description

File specification for a Sql database.

Windows Installer references

None

Parents

SqlDatabase

Inner Text

None

Children

None

Attributes

Name	Туре	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 GB, MB and KB and % suffixes can be used to specify gigabytes, megabytes, kilobytes or a percentage of the current file size to grow. The default is megabytes if no suffix is specified. The default value is 10% if GrowthSize is not specified, and the minimum value is 64 KB. 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. The GB, MB and KB suffixes can be used to to specify gigabytes, megabytes or kilobytes. The default is megabytes if no suffix is specified. If MaxSize is not specified, the file will grow until the disk is full.
Name	String	Specifies the logical name for the log file.
Size	String	Specifies the size of the log file. The GB, MB and KB suffixes can be used to specify gigabytes, megabytes or kilobytes. The default is megabytes if no suffix is specified. When a Size is not supplied for a log file, SQL Server makes the file 1 MB.

See Also

Sql Schema

Version 3.5.1623.0

SqlScript Element (Sql Extension)

Description

SQL Script

Windows Installer references

None

Parents

Component, SqlDatabase

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String		Yes
BinaryKey	String	Reference to Binary stream that contains the SQL script to execute.	Yes
ContinueOnError	YesNoType	Continue executing scripts even if this one fails.	
ExecuteOnInstall	YesNoType	Specifies to execute the script when the associated component is installed. This attribute is mutually exclusive with the RollbackOnInstall,	

RollbackOnReinstall and RollbackOnUninstall attributes.

ExecuteOnReinstall <u>YesNoType</u>

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. This attribute is mutually exclusive with the RollbackOnInstall. RollbackOnReinstall

ExecuteOnUninstall <u>YesNoType</u>

Specifies to execute the script when the associated component is uninstalled. This attribute is mutually exclusive with the RollbackOnInstall, RollbackOnReinstall and RollbackOnUninstall

RollbackOnUninstall

and

attributes.

attributes.

RollbackOnInstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to install the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall and ExecuteOnUninstall attributes.
RollbackOnReinstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to reinstall the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall and ExecuteOnUninstall attributes.
RollbackOnUninstall	YesNoType	Specifies whether to execute the script on rollback if an attempt is made to uninstall the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall

		and ExecuteOnUninstall attributes.
Sequence	Integer	Specifes 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

Sql Schema

Version 3.5.1623.0

SqlString Element (Sql Extension)

Description

SQL String

Windows Installer references

None

Parents

Component, SqlDatabase

Inner Text

None

Children

None

Attributes

Name	Туре	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. This attribute is mutually exclusive with the RollbackOnInstall, RollbackOnReinstall and RollbackOnUninstall attributes.	

ExecuteOnReinstall <u>YesNoType</u> 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. This

attribute is mutually

exclusive with the RollbackOnInstall,

RollbackOnReinstall

and

RollbackOnUninstall

attributes.

<u>YesNoType</u> ExecuteOnUninstall Specifies to execute

the string when the

associated

component is

uninstalled. This

attribute is mutually exclusive with the

RollbackOnInstall,

RollbackOnReinstall

and

RollbackOnUninstall

attributes.

RollbackOnInstall

YesNoType Specifies whether to

execute the string on

rollback if an attempt

		is made to install the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall and ExecuteOnUninstall attributes.
RollbackOnReinstall	YesNoType	Specifies whether to execute the string on rollback if an attempt is made to reinstall the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall and ExecuteOnUninstall attributes.
RollbackOnUninstall	YesNoType	Specifies whether to execute the string on rollback if an attempt is made to uninstall the associated component. This attribute is mutually exclusive with the ExecuteOnInstall, ExecuteOnReinstall and ExecuteOnUninstall attributes.
Sequence	Integer	Specifes 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

Sql Schema

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'.$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]=([.\%])=([.\%])$ [_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z.]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Util 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 Utility Extension.

Target Namespace

http://schemas.microsoft.com/wix/UtilExtension

Child Elements

- CloseApplication
- ComponentSearch
- <u>DirectorySearch</u>
- EventManifest
- EventSource
- FileSearch
- FileShare
- FileSharePermission
- Group
- GroupRef
- InternetShortcut
- PerfCounter
- PerfCounterManifest
- PerformanceCategory
- PerformanceCounter
- PermissionEx
- ProductSearch

- RegistrySearch
- ServiceConfig
- <u>User</u>
- XmlConfig
- XmlFile

CloseApplication Element (Util Extension)

Description

Closes applications or schedules a reboot if application cannot be closed.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text (xs:string)

Condition that determines if the application should be closed. Must be blank or evaluate to true for the application to be scheduled for closing.

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Identifier for the close application (primary key).	Yes
CloseMessage	YesNoType	Optionally sends a close message to the application. Default is no.	
Description	String	Description to show if application is running and	

		needs to be closed.	
ElevatedCloseMessage	YesNoType	Optionally sends a close message to the application from deffered action without impersonation. Default is no.	
Property	String	Property to be set if application is still running. Useful for launch conditions or to conditionalize custom UI to ask user to shut down apps.	
RebootPrompt	YesNoType	Optionally prompts for reboot if application is still running. Default is yes.	
Sequence	Integer	Optionally orders the applications to be closed.	
Target	String	Name of the exectuable to be closed. This should only be file name.	Yes

See Also

Util Schema

ComponentSearch Element (Util Extension)

Description

Describes a component search.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
After	String	Id of the search that this one should come after.	
Condition	String	Condition for evaluating the search. If this evaluates to false, the search is not executed at all.	
Guid	String	Component to search for.	Yes
Id	String	Id of the search for ordering and dependency.	
ProductCode	String	Optional ProductCode to determine if the component is installed.	

) DCI	ı	lt
\Box	にこり	ш	

Enumeration Rather than saving the matching key path into the variable, a ComponentSearch can save an attribute of the component instead. This attribute's value must be one of the following: Directory

Saves the parent directory for the component's file key path; other types of key path are returned unmodified.

State

Saves the state of the component: absent (2), locally installed (3), will run from source (4), or installed in default location (either local or from source) (5)

KeyPath

Saves the key path of the component if installed. This is the default.

Variable

String

Name of the variable in which to place the result of Yes

the search.

See Also

Util Schema

DirectorySearch Element (Util Extension)

Description

Describes a directory search.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
After	String	Id of the search that this one should come after.	
Condition	String	Condition for evaluating the search. If this evaluates to false, the search is not executed at all.	
ld	String	Id of the search for ordering and dependency.	
Path	String	Directory path to search for.	
Result	NMTOKEN	Rather than saving the matching directory path into the variable, a DirectorySearch can save an attribute of the matching directory instead. Pattern:	

		'Exists'.	
Variable	String	Name of the variable in which to place the result of the search.	Yes

See Also

Util Schema

EventManifest Element (Util Extension)

Description

Used to install Event Manifests.

Windows Installer references

None

Parents

<u>File</u>

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
MessageFile	String	The message file (including path) of all the providers in the event manifest. Often the message file path cannot be determined until setup time. Put your MessageFile here and the messageFileName attribute of the all the providers in the manifest will be updated with the path before it is registered.	
ParameterFile	String	The parameter file (including path) of all the providers in the event manifest. Often the parameter file path cannot be determined until setup time. Put	

your ParameterFile here and the parameterFileName attribute of the all the providers in the manifest will be updated with the path before it is registered.

ResourceFile String

The resource file (including path) of all the providers in the event manifest. Often the resource file path cannot be determined until setup time. Put your ResourceFile here and the resourceFileName attribute of the all the providers in the manifest will be updated with the path before it is registered.

See Also

Util Schema

EventSource Element (Util Extension)

Description

Creates an event source.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Attributes

Name	Туре	Description
CategoryCount	Integer	The number of categories in CategoryMessageFile. CategoryMessageFile must be specified too.
CategoryMessageFile	String	Name of the category messag CategoryCount must be specifitoo. Note that this is a formatte field, so you can use [#fileId] s to refer to a file being installed also written as a REG_EXPAN string, so you can use %environment_variable% syntrefer to a file already present cuser's machine.

String	Name of the event message fi Note that this is a formatted fig you can use [#fileId] syntax to to a file being installed. It is als written as a REG_EXPAND_S string, so you can use %environment_variable% synt refer to a file already present couser's machine.
YesNoType	Marks the EventSource registreshe key path of the component belongs to.
String	Name of the event source's lo
String	Name of the event source.
String	Name of the parameter messa file. Note that this is a formatte field, so you can use [#fileId] s to refer to a file being installed also written as a REG_EXPAN string, so you can use %environment_variable% synt refer to a file already present of user's machine.
YesNoType	Equivalent to EVENTLOG_ERROR_TYPE.
<u>YesNoType</u>	Equivalent to EVENTLOG_AUDIT_FAILURI
YesNoType	Equivalent to EVENTLOG_INFORMATION_
<u>YesNoType</u>	Equivalent to EVENTLOG_AUDIT_SUCCES
YesNoType	Equivalent to EVENTLOG_WARNING_TYP
	YesNoType String String String YesNoType YesNoType YesNoType

See Also <u>Util Schema</u>

FileSearch Element (Util Extension)

Description

Describes a file search.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
After	String	Id of the search that this one should come after.	
Condition	String	Condition for evaluating the search. If this evaluates to false, the search is not executed at all.	
Id	String	Id of the search for ordering and dependency.	
Path	String	File path to search for.	
Result	Enumeration	Rather than saving the matching file path into the variable, a FileSearch can save an attribute of the matching file instead. This attribute's value must be one	

of the following:

Exists

Saves true if a matching file is found; false otherwise.

Version

Saves the version information for files that have it (.exe, .dll); zeroversion (0.0.0.0) otherwise.

Variable String Name of the variable in which Yes to place the result of the search.

See Also

Util Schema

FileShare Element (Util Extension)

Description

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

Windows Installer references

None

Parents

Component

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. FileSharePermission (min: 1, max: unbounded): ACL permission

Attributes

Name	Туре	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

Util Schema

FileSharePermission Element (Util Extension)

Description

Sets ACLs on a FileShare. This element has no Id attribute. The table and key are taken from the parent element.

Windows Installer references

None

Parents

FileShare

Inner Text

None

Children

None

Name	Туре	Description
ChangePermission	YesNoType	
CreateChild	YesNoType	For a directory, the right to cresubdirectory. Only valid under 'CreateFolder' parent.
CreateFile	YesNoType	For a directory, the right to crefile in the directory. Only valid under a 'CreateFolder' parent
Delete	<u>YesNoType</u>	
DeleteChild	YesNoType	For a directory, the right to de directory and all the files it contains, including read-only Only valid under a 'CreateFolion's

parent.

GenericAll	YesNoType	
GenericExecute	YesNoType	
GenericRead	YesNoType	specifying this will fail to grant access
GenericWrite	<u>YesNoType</u>	
Read	YesNoType	
ReadAttributes	<u>YesNoType</u>	
ReadExtendedAttributes	YesNoType	
ReadPermission	YesNoType	
Synchronize	<u>YesNoType</u>	
TakeOwnership	<u>YesNoType</u>	
Traverse	YesNoType	For a directory, the right to tra the directory. By default, users assigned the BYPASS_TRAVERSE_CHEC privilege, which ignores the FILE_TRAVERSE access right Only valid under a 'CreateFolioparent.
User	String	
WriteAttributes	YesNoType	
WriteExtendedAttributes	YesNoType	

See Also

Util Schema

Group Element (Util Extension)

Description

Finds user groups on the local machine or specified Active Directory domain. The local machine will be searched for the group first then fallback to looking in Active Directory. This element is not capable of creating new groups but can be used to add new or existing users to an existing group.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Unique identifier in your installation package for this group.	Yes
Domain	String	An optional <u>Formatted</u> string that specifies the domain for the group.	
Name	String	A <u>Formatted</u> string that contains the name of the group to be found.	Yes

See Also

Util Schema

GroupRef Element (Util Extension)

Description

Used to join a user to a group

Windows Installer references

None

Parents

<u>User</u>

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes

See Also

Util Schema

InternetShortcut Element (Util Extension)

Description

Creates a shortcut to a URL.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Name	Туре	Description	Required
ld	String	Unique identifier in your installation package for this Internet shortcut.	Yes
Directory	String	Identifier reference to Directory element where shortcut is to be created. This attribute's value defaults to the parent Component directory.	
Name	String	The name of the shortcut file, which is visible to the user. (The .lnk extension is added automatically and by default, is not shown to the user.)	Yes
Target	String	URL that should be opened	Yes

when the user selects the shortcut. Windows opens the URL in the appropriate handler for the protocol specified in the URL. Note that this is a formatted field, so you can use [#fileId] syntax to refer to a file being installed (using the file: protocol).

Type

Enumeration Which type of shortcut should be created. This attribute's value must be one of the following: url

> Creates .url files using IUniformResourceLocatorW.

link

Creates .Ink files using IShellLinkW (default).

How Tos and Examples

How To: Create a shortcut to a webpage

See Also

Util Schema

PerfCounter Element (Util Extension

Description

This element has been deprecated; please use the <u>PerformanceCounter</u> element instead.

Windows Installer references

None

Parents

File

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Name	String		

See Also

Util Schema

PerfCounterManifest Element (Util Extension)

Description

Used to install Perfmon Counter Manifests.

Windows Installer references

None

Parents

File

Inner Text

None

Children

None

Name	Туре	Description	Required
ResourceFileDirectory	String	The directory that holds the resource file of the providers in the perfmon counter manifest. Often the resource file path cannot be determined until setup time. Put the directory here and during perfmon manifest registrtion the path will be updated in the registry. If not specified, Perfmon will look for the resource	

file in the same directory of the perfmon counter manifest file.

See Also

Util Schema

PerformanceCategory Element (Util Extension)

Description

Used to create performance categories and configure performance counters.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. PerformanceCounter (min: 0, max: unbounded)

Name	Туре	Description
Close	String	Function ent the Library I when closing performance default is "ClosePerfor which should all managed performance
Collect	String	Function ent the Library E when collect the performa

		The default i "CollectPerformhich should all managed performance
DefaultLanguage	<u>PerformanceCounterLanguageType</u>	Default lang performance and containe names and I
Help	String	Optional helperformance category.
Id	String	Unique identinstallation pthis performation category.
Library	String	DLL that cor performance default is "ne which should all managed performance
MultiInstance	YesNoType	Flag that spe whether the counter cate or single ins Default is sir
Name	String	Name for the performance category. If t not provided attribute is u name of the counter cate
Open	String	Function ent

the Library E when openir performance default is "OpenPerfor which should all managed performance

See Also <u>Util Schema</u>

PerformanceCounter Element (Util Extension)

Description

Creates a performance counter in a performance category.

Windows Installer references

None

Parents

PerformanceCategory

Inner Text

None

Children

None

Name	Туре	Description
Help	String	Optional help text for the performance counter.
Language	PerformanceCounterLanguageType	Language for the peformance counte name and help. The default is to use the parent PerformanceCategorelement's DefaultLanguage attribute.
Name	String	Name for the performance counte

Type PerformanceCounterTypesType Type of the performance counts

See Also <u>Util Schema</u>

PermissionEx Element (Util Extension)

Description

Sets ACLs on File, Registry, CreateFolder, or ServiceInstall. 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. To use PermissionEx with an IA-64 MSI, you must compile all of your source files with the "-arch ia64" switch, to ensure the IA-64 custom action is used, and not the x64 custom action.

Windows Installer references

None

Parents

CreateFolder, File, Registry, RegistryKey, RegistryValue

Inner Text

None

Children

None

Name	Туре	Description
Append	<u>YesNoType</u>	
ChangePermission	YesNoType	
CreateChild	YesNoType	For a directory, the righ subdirectory. Only valid 'CreateFolder' parent.
CreateFile	<u>YesNoType</u>	For a directory, the righ file in the directory. Only under a 'CreateFolder'

CreateLink	<u>YesNoType</u>	
CreateSubkeys	<u>YesNoType</u>	
Delete	<u>YesNoType</u>	
DeleteChild	YesNoType	For a directory, the righ directory and all the file contains, including read Only valid under a 'Creparent.
Domain	String	
EnumerateSubkeys	<u>YesNoType</u>	
Execute	<u>YesNoType</u>	
GenericAll	<u>YesNoType</u>	
GenericExecute	<u>YesNoType</u>	
GenericRead	<u>YesNoType</u>	specifying this will fail to access
GenericWrite	YesNoType	
Notify	<u>YesNoType</u>	
Read	<u>YesNoType</u>	
ReadAttributes	<u>YesNoType</u>	
ReadExtendedAttributes	<u>YesNoType</u>	
ReadPermission	<u>YesNoType</u>	
ServiceChangeConfig	YesNoType	Required to call the ChangeServiceConfig of ChangeServiceConfig2 change the service confolly valid under a 'Serparent.
ServiceEnumerateDependents	YesNoType	Required to call the EnumDependentServic

		to enumerate all the se dependent on the servi- valid under a 'ServiceIn parent.
ServiceInterrogate	YesNoType	Required to call the Co function to ask the serv its status immediately. (under a 'ServiceInstall'
ServicePauseContinue	YesNoType	Required to call the Co function to pause or co service. Only valid unde 'ServiceInstall' parent.
ServiceQueryConfig	YesNoType	Required to call the QueryServiceConfig an QueryServiceConfig2 for query the service configured only valid under a 'Serparent.
ServiceQueryStatus	YesNoType	Required to call the QueryServiceStatus fur the service control man the status of the service under a 'ServiceInstall'
ServiceStart	YesNoType	Required to call the Sta function to start the ser valid under a 'ServiceIn parent.
ServiceStop	<u>YesNoType</u>	Required to call the Co function to stop the servalid under a 'ServiceIn parent.
ServiceUserDefinedControl	YesNoType	Required to call the Co function to specify a us control code. Only valic 'ServiceInstall' parent.

Synchronize	<u>YesNoType</u>	
TakeOwnership	YesNoType	
Traverse	YesNoType	For a directory, the righ the directory. By default assigned the BYPASS_TRAVERSE_ privilege, which ignores FILE_TRAVERSE acce Only valid under a 'Cresparent.
User	String	
Write	YesNoType	
WriteAttributes	YesNoType	
WriteExtendedAttributes	YesNoType	

See Also

Util Schema

ProductSearch Element (Util Extension)

Description

Describes a product search.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

None

Name	Туре	Description	Required
After	String	Id of the search that this one should come after.	
Condition	String	Condition for evaluating the search. If this evaluates to false, the search is not executed at all.	
Guid	String	ProductCode to search for.	Yes
Id	String	Id of the search for ordering and dependency.	
Result	Enumeration	Rather than saving the product version into the variable, a ProductSearch can save another attribute of the	

matching product instead. This attribute's value must be one of the following:

Version

Saves the version of a matching product if found; 0.0.0.0 otherwise. This is the default.

Language

Saves the language of a matching product if found; empty otherwise.

State

Saves the state of the product: advertised (1), absent (2), or locally installed (3).

Assignment

Saves the assignment type of the product: peruser (0), or per-machine (1).

Variable	String
----------	--------

Name of the variable in which Yes to place the result of the search.

See Also

Util Schema

RegistrySearch Element (Util Extension)

Description

Describes a registry search.

Windows Installer references

None

Parents

Bundle, Fragment

Inner Text

None

Children

None

Name	Туре	Description
After	String	Id of the search that thi should come after.
Condition	String	Condition for evaluating search. If this evaluates false, the search is not executed at all.
ExpandEnvironmentVariables	YesNoType	Whether to expand any environment variables i REG_SZ, REG_EXPAN or REG_MULTI_SZ val
Format	Enumeration	What format to return the value in. This attribute's must be one of the followard.

		Returns the unforn value directly from registry. For example REG_DWORD value is returned as '1', r Compatible Returns the value formatted as Windelinstaller would. Fo example, a REG_DWORD value is returned as '#1',
Id	String	Id of the search for ordeand dependency.
Key	String	Key to search for.
Result	NMTOKEN	Rather than saving the matching registry value the variable, a Registry can save an attribute o matching entry instead Pattern: 'Exists'.
Root	Enumeration	Registry root hive to se under. This attribute's v must be one of the follo HKLM HKEY_LOCAL_M/ HKCU HKEY_CURRENT HKCR HKEY_CLASSES_ HKU HKEY_USERS
Value	String	Optional value to searc

Variable String Name of the variable in to place the result of the search.			under the given Key.
	Variable	String	to place the result of the

See Also

Util Schema

ServiceConfig Element (Util Extension)

Description

Service configuration information for failure actions.

Windows Installer references

None

Parents

Component, ServiceInstall

Inner Text

None

Children

None

Name	Туре	Description	Req
FirstFailureActionType	Enumeration	Action to take on the first failure of the service. This attribute's value must be one of the following: none	Yes
		reboot	
		restart	
		runCommand	
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	Yes

		on the second failure of the service. This attribute's value must be one 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 must be one of the following: none	Yes
		reboot	
		restart	
		runCommand	

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 <u>Util Schema</u>

User Element (Util Extension)

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, Module, Product

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. GroupRef (min: 0, max: unbounded)

Name	Туре	Description
Id	String	
CanNotChangePassword	YesNoType	The user cannot change the account's password. Equival to UF_PASSWD_CANT_CHAN
CreateUser	YesNoType	Indicates whether or not to create the user. User creatio can be skipped if all that is desired is to join a user to groups.

Disabled	YesNoType	The account is disabled. Equivalent to UF_ACCOUNTDISABLE.
Domain	String	A Formatted string that conta the local machine or Active Directory domain for the use
FailIfExists	<u>YesNoType</u>	Indicates if the install should if the user already exists.
LogonAsService	YesNoType	Indicates whether or not the can logon as a serivce. User creation can be skipped if all is desired is to set this acces right on the user.
Name	String	A Formatted string that conta the name of the user accoun
Password	String	Usually a Property that is passed in on the command-l to keep it more secure.
PasswordExpired	YesNoType	Indicates whether the user m change their password on th first login.
PasswordNeverExpires	YesNoType	The account's password nev expires. Equivalent to UF_DONT_EXPIRE_PASSV
RemoveOnUninstall	YesNoType	Indicates whether the user account should be removed left behind on uninstall.
UpdateIfExists	YesNoType	Indicates if the user account properties should be updated the user already exists.

See Also

Util Schema, Group, GroupRef

XmlConfig Element (Util Extension)

Description

Adds or removes .xml file entries. If you use the XmlConfig element you must reference WixUtilExtension.dll as it contains the XmlConfig custom actions.

Windows Installer references

None

Parents

Component, XmlConfig

Inner Text (xs:string)

This element may have inner text.

Children

Sequence (min: 1, max: 1)

1. XmlConfig (min: 0, max: unbounded)

Name	Туре	Description	Required
Id	String	Identifier for xml file modification.	Yes
Action	Enumeration	This attribute's value must be one of the following: create delete	
ElementId	String	The Id of another XmlConfig to add attributes to. In this case, the	

		'Action' 'Node' and 'On' attributes must be left unspecified.	
ElementPath	String	The XPath of the parent element being modified. Note that this is a formatted field and therefore, square brackets in the XPath must be escaped.	
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.	
Node	Enumeration	This attribute's value must be one of the following: element value	
		document	

On	Enumeration	This attribute's value must be one of the following: install uninstall
PreserveModifiedDate	YesNoType	Specifies wheter or not the modification should preserve the modified date. Preserving the modified date will allow the file to be patched if no other modifications have been made.
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 the Formatted topic for information how to escape square brackets

		in the value.	
VerifyPath	String	The XPath to the element being modified. This is required for 'delete' actions. For 'create' actions, VerifyPath is used to decide if the element already exists.	

See Also

Util Schema

XmlFile Element (Util Extension)

Description

Adds or removes .xml file entries. If you use the XmlFile element you must reference WixUtilExtension.dll as it contains the XmlFile custom actions.

Windows Installer references

None

Parents

Component

Inner Text

None

Children

None

Name	Туре	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 must be one of the following: createElement Creates a new element under the element	Yes

specified in ElementPath. The Name attribute is required in this case and specifies the name 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.

bulkSetValue Sets all the values in the

elements that match the ElementPath. If Name is specified, attributes with that name are set to the same value specified in Value. If Name is not specified, the text values of the elements are set. Value is a required attribute if setBulkValue is the action specified.

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.
PreserveModifiedDate	YesNoType	Specifies wheter or not the modification should preserve the modified date. Preserving the modified date will allow the file to be patched if no other modifications have been made.
SelectionLanguage	Enumeration	Specify whether the DOM object should use XPath language or the old XSLPattern language (default) as the query language. This attribute's value

		must be one of the following: XPath XSLPattern
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 the Formatted topic for information how to escape square brackets in the value.

See Also

Util Schema

AutogenGuid (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 setting the value to "*". 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]\{12\}[\})]?|[\{(]?\?\{8\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\-?\{4\}\$

See Also

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\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)|*|^\$'.

See Also

ExitType (Simple Type)

Description

TODO: What does this type mean?

Enumeration Type

Possible values: {success, cancel, error, suspend}

See Also

Guid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}". 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}[})]?|PUT\-GUID\-(?:\d+\-)?HERE|([!\$])(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_.]*\)'.

See Also

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

InstallUninstallType (Simple Type)

Description

Specifies whether an action occur on install, uninstall or both.

Enumeration Type

Possible values: {install, uninstall, both}

See Also

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) where "Variable" is the name of the variable.

Pattern Type

Must match the regular expression: $'[0-9][0-9]*|([!$])\setminus((?:loc|bind)\setminus A-Za-z][0-9A-Za-z_.]+\)'.$

See Also

LongFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]{1,259}|([!$])\(loc\. [_A-Za-z][0-9A-Za-z_.]*\)'.$

See Also

PerformanceCounterLanguageType (Simple Type)

Description

Enumeration of valid languages for performance counters.

Enumeration Type

Possible values: {afrikaans, albanian, arabic, armenian, assamese, azeri, basque, belarusian, bengali, bulgarian, catalan, chinese, croatian, czech, danish, divehi, dutch, english, estonian, faeroese, farsi, finnish, french, galician, georgian, german, greek, gujarati, hebrew, hindi, hungarian, icelandic, indonesian, italian, japanese, kannada, kashmiri, kazak, konkani, korean, kyrgyz, latvian, lithuanian, macedonian, malay, malayalam, manipuri, marathi, mongolian, nepali, norwegian, oriya, polish, portuguese, punjabi, romanian, russian, sanskrit, serbian, sindhi, slovak, slovenian, spanish, swahili, swedish, syriac, tamil, tatar, telugu, thai, turkish, ukrainian, urdu, uzbek, vietnamese}

See Also

PerformanceCounterTypesType (Simple Type)

Description

Enumeration of valid types for performance counters.

Enumeration Type

Possible values: {averageBase, averageCount64, averageTimer32, counterDelta32, counterTimerInverse, sampleFraction, timer100Ns, counterTimer, rawFraction, timer100NsInverse, counterMultiTimer100Ns, counterMultiTimerInverse, counterMultiTimer100NsInverse, elapsedTime, sampleBase, rawBase, counterMultiBase, rateOfCountsPerSecond64, rateOfCountsPerSecond32, countPerTimeInterval64, countPerTimeInterval32, sampleCounter, counterDelta64, numberOfItems64, numberOfItems32, numberOfItemsHEX64, numberOfItemsHEX32}

See Also

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

SequenceType (Simple Type)

Description

Controls which sequences the item assignment is sequenced in.

Enumeration Type

Possible values: {both, execute, ui}

See Also

ShortFileNameType (Simple Type)

Description

Values of this type will look like: "FileName.ext". Only one period is allowed. 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: $'[^{\}] <: \Lambda^*''+,;=[]].]{1,8}(... [^{\}] <: \Lambda^*''+,;=[]].]{0,3})?|([!$])(loc).[A-Za-z][0-9A-Za-z .]*\)'.$

See Also

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

WildCardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File N?me.extension*". Legal long names contain no more than 260 characters and must contain at least one non-period character. 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

WildCardShortFileNameType (Simple Type)

Description

Pattern Type

Must match the regular expression: $'[^{\}]<:/''+,;=\[]\] \{1,16\}(\.[^{\}]<:/''+,;=\[]\] \{0,6\})?|([!$])(loc\.[A-Za-z][0-9A-Za-z].]*\)'.$

See Also

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

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/VSExtension

Child Elements

- HelpCollection
- HelpCollectionRef
- <u>HelpFile</u>
- HelpFileRef
- HelpFilter
- <u>HelpFilterRef</u>
- PlugCollectionInto

HelpCollection Element (Vs Extension)

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	Туре	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
SuppressCustomActions	YesNoType	Suppress linking Help registration	

custom actions.
Help
redistributable
merge modules
will be required.
Use this when
building a merge
module.

See Also

Vs Schema

HelpCollectionRef Element (Vs Extension)

Description

Create a reference to a HelpCollection element in another Fragment.

Windows Installer references

None

Parents

Fragment, Module, 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

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Vs Schema

HelpFile Element (Vs Extension)

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.	Yes
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.
SuppressCustomActions	YesNoType	Suppress linking Help registration custom actions. Help redistributable merge modules will be required. Use this when building a merge module.

See Also

Vs Schema

HelpFileRef Element (Vs Extension)

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	Туре	Description	Required
Id	String	Primary Key for HelpFile Table.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Vs Schema

HelpFilter Element (Vs Extension)

Description

Filter for Help Namespace.

Windows Installer references

None

Parents

Fragment, Module, Product

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Primary Key for HelpFilter.	Yes
FilterDefinition	String	Query String for Help Filter.	
Name	String	Friendly name for Filter.	Yes
SuppressCustomActions	YesNoType	Suppress linking Help registration custom actions. Help redistributable merge modules will be required. Use this when building a merge module.	

See Also

Vs Schema

HelpFilterRef Element (Vs Extension)

Description

Create a reference to a HelpFile element in another Fragment.

Windows Installer references

None

Parents

HelpCollection

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Required
Id	String	Primary Key for HelpFilter.	Yes

Any Attribute (namespace='##other' processContents='lax') Extensibility point in the WiX XML Schema. Schema extensions can register additional attributes at this point in the schema.

See Also

Vs Schema

PlugCollectionInto Element (Vs Extension)

Description

Plugin for Help Namespace.

Windows Installer references

None

Parents

HelpCollection

Inner Text

None

Children

None

Attributes

Name	Туре	Description	Rec
Attributes	String	Key for HxA (Attributes) file of child namespace.	
SuppressExternalNamespaces	YesNoType	Suppress linking Visual Studio Help namespaces. Help redistributable merge modules will be required. Use this when building a merge module.	

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. The following special keys can be used to plug into external namespaces defined outside of the installer. MS_VSIPCC_v80: Visual Studio 2005 MS.VSIPCC.v90: Visual Studio 2008	Yes
TargetFeature	String	Key for the feature parent of this help collection. Required only when plugging into external namespaces.	
TargetTableOfContents	String	Key for HxT file of parent namespace that now includes the new child namespace.	

See Also Vs Schema

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

Vs Schema

Developing for WiX

This section covers the following topics for developers who want to contribute to the WiX code base:

- How to be a Windows Installer XML Developer
- **Building WiX**
- NAnt Conventions
- Extension Development
- Developing for Votive
- Adding to the WiX Documentation
- Testing WiX

So you want to be a Windows Installer XML developer?

People have started expressing interest in joining the <u>Windows Installer XML toolset</u> 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 <u>Windows Installer XML mailing lists</u> 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. :)

Reprinted from http://blogs.msdn.com/robmen/archive/2004/04/14/112970.aspx. Copyright **P** Rob Mensching

Building WiX

Simply run "nant" from the dev\wix directory. This will build debug bits into the "target" directory by default. To build release bits, run "nant - D:flavor=ship". You can disable building IA64-specific parts of the custom action library by running "nant -D:ia64=false".

In order to fully build WiX, you must have the following Frameworks and SDKs installed:

- NAnt (build 2008-02-10-0.86 or later)
- The following components from the <u>Windows SDK for Windows</u> <u>Server 2008 and .NET Framework 3.5</u>, Visual Studio 2008, Microsoft Windows 7 SDK, and/or Visual Studio 2010:
 - x86 and x64 compilers, headers and libraries
 - IA64 headers and libraries are optional, but they are necessary for IA64 custom action support
 - If you want to be able to build optimized IA64 binaries, you'll need both the Windows SDK for Windows Server 2008 and .NET Framework 3.5 SDK AND Visual Studio 2008 installed; or the Microsoft Windows 7 SDK AND Visual Studio 2010.
- <u>HTML Help SDK 1.4</u> or higher [installed to Program Files or Program Files (x86)]

To build Sconce and Votive, you must have the following SDKs installed:

- Visual Studio 2005 SDK Version 4.0
- Visual Studio 2008 SDK

More information about the Visual Studio SDK can be found at the <u>Visual Studio Extensibility Center</u>.

To install Votive on Visual Studio 2005, 2008, or 2010, you must have the Standard Edition or higher.

To successfully build WiX with only Windows Server 2008 and .NET Framework 3.5 SDK (without Visual Studio 2008), you need to modify your NAnt.exe.config file to support the Windows Server 2008 and .NET Framework 3.5 SDK.

```
<readregistry
property="sdkInstallRoot"
key="SOFTWARE\Microsoft\Microsoft SDKs\Windows\v6.0a\WinSDKNetFx
hive="LocalMachine"
failonerror="false"/>
```

Replace this with the following element:

```
<readregistry
property="sdkInstallRoot"
key="SOFTWARE\Microsoft\Microsoft SDKs\Windows\v6.1\WinSDKNetFxT
hive="LocalMachine"
failonerror="false"/>
```

Note the only difference is that the "v6.0a" changed to "v6.1" in the "key" attribute.

To successfully build WiX with only the Microsoft Windows 7 SDK and/or Visual Studio 2010, you need to modify your NAnt.exe.config file to support either and the .NET Framework 4.0. See NAnt Configuration for an example.

To build DTF help files, you need the following tools:

- Sandcastle January 2008 Release
- Sandcastle Help File Builder 1.6.0.4

The DTF help build looks for them in an "external" directory parallel to the WiX "src" directory:

- Sandcastle January 2008 Release: external\Sandcastle
- Sandcastle Help File Builder 1.6.0.4: external\SandcastleBuilder

NAnt Configuration

Microsoft .NET Framework 4.0 Support

In order to build WiX using NAnt with only Microsoft Visual Studio 2010 and the .NET Framework 4.0, you need to add the following to the list of frameworks supported by NAnt.

```
<framework name="net-4.0" family="net" version="4.0" description="Microsof</pre>
 <!-- ${installVersion} -->
 <runtime>
  probing-paths>
   <directory name="lib/net/4.0" />
   <directory name="lib/net/neutral" />
   <directory name="lib/common/4.0" />
   <directory name="lib/common/neutral" />
  </probing-paths>
  <modes>
   <strict>
    <environment>
     <variable name="COMPLUS_VERSION" value="v${installVersion}" />
    </environment>
   </strict>
  </modes>
 </runtime>
 <reference-assemblies basedir="${path::combine(installRoot, 'v' + ${installVer}
  <include name="Accessibility.dll" />
  <include name="mscorlib.dll" />
  <include name="Microsoft.Build.Utilities.dll" />
  <include name="Microsoft.Vsa.dll" />
  <include name="Microsoft.VisualBasic.dll" />
  <include name="Microsoft.VisualBasic.Compatibility.dll" />
  <include name="Microsoft.VisualBasic.Compatibility.Data.dll" />
  <include name="System.Configuration.dll" />
  <include name="System.Configuration.Install.dll" />
```

```
<include name="System.Data.dll" />
<include name="System.Data.OracleClient.dll" />
<include name="System.Data.SqlXml.dll" />
<include name="System.Deployment.dll" />
<include name="System.Design.dll" />
<include name="System.DirectoryServices.dll" />
<include name="System.dll" />
<include name="System.Drawing.Design.dll" />
<include name="System.Drawing.dll"/>
<include name="System.EnterpriseServices.dll" />
<include name="System.Management.dll" />
<include name="System.Messaging.dll"/>
<include name="System.Runtime.Remoting.dll" />
<include name="System.Runtime.Serialization.Formatters.Soap.dll" />
<include name="System.Security.dll" />
<include name="System.ServiceProcess.dll" />
<include name="System.Transactions.dll" />
<include name="System.Web.dll" />
<include name="System.Web.Mobile.dll" />
<include name="System.Web.RegularExpressions.dll" />
<include name="System.Web.Services.dll" />
<include name="System.Windows.Forms.dll" />
<include name="System.Xml.dll" />
<include name="Microsoft.Build.Engine.dll" />
<include name="Microsoft.Build.Framework.dll" />
<include name="System.AddIn.Contract.dll" />
<include name="System.AddIn.dll" />
<include name="System.Core.dll" />
<include name="System.Data.DataSetExtensions.dll" />
<include name="System.Data.Linq.dll" />
<include name="System.DirectoryServices.AccountManagement.dll" />
<include name="System.Management.Instrumentation.dll" />
<include name="System.Net.dll" />
<include name="System.ServiceModel.Web.dll" />
<include name="System.Web.Extensions.Design.dll" />
<include name="System.Web.Extensions.dll" />
<include name="System.Windows.Presentation.dll" />
<include name="System.WorkflowServices.dll" />
<include name="System.Xml.Ling.dll" />
```

```
<include name="System.IdentityModel.dll" />
 <include name="System.IdentityModel.Selectors.dll" />
 <include name="System.IO.Log.dll" />
 <include name="System.Printing.dll"/>
 <include name="System.Runtime.Serialization.dll" />
 <include name="System.ServiceModel.dll" />
 <include name="System.Speech.dll" />
 <include name="System.Workflow.Activities.dll" />
 <include name="System.Workflow.ComponentModel.dll" />
 <include name="System.Workflow.Runtime.dll" />
 <include name="WindowsBase.dll" />
</reference-assemblies>
<task-assemblies>
 <!-- include MS.NET version-neutral assemblies -->
 <include name="extensions/net/neutral/**/*.dll" />
 <!-- include MS.NET 4.0 specific assemblies -->
 <include name="extensions/net/4.0/**/*.dll" />
 <!-- include MS.NET specific task assembly -->
 <include name="NAnt.MSNetTasks.dll" />
 <!-- include MS.NET specific test assembly -->
 <include name="NAnt.MSNet.Tests.dll" />
 <!-- include .NET 4.0 specific assemblies -->
 <include name="extensions/common/4.0/**/*.dll" />
</task-assemblies>
<tool-paths>
 <directory name="${path::combine(sdkInstallRoot, 'bin')}" if="${property::ex</pre>
 <directory name="${installRoot}" />
</tool-paths>
ct>
 <readregistry property="installRoot" key="SOFTWARE\Microsoft\NET Fran
 <readregistry property="installVersion" key="SOFTWARE\Microsoft\NET Fi</pre>
 <readregistry property="sdkInstallRoot" key="SOFTWARE\Microsoft\Micros
 <readregistry property="sdkInstallRoot" unless="${property::exists('sdkInstal
</project>
<tasks>
 <task name="csc">
  <attribute name="supportsnowarnlist">true</attribute>
  <attribute name="supportswarnaserrorlist">true</attribute>
  <attribute name="supportskeycontainer">true</attribute>
```

```
<attribute name="supportskeyfile">true</attribute>
   <attribute name="supportsdelaysign">true</attribute>
   <attribute name="supportsplatform">true</attribute>
   <attribute name="supportslangversion">true</attribute>
  </task>
  <task name="vbc">
   <attribute name="supportsdocgeneration">true</attribute>
   <attribute name="supportsnostdlib">true</attribute>
   <attribute name="supportsnowarnlist">true</attribute>
   <attribute name="supportskeycontainer">true</attribute>
   <attribute name="supportskeyfile">true</attribute>
   <attribute name="supportsdelaysign">true</attribute>
   <attribute name="supportsplatform">true</attribute>
   <attribute name="supportswarnaserrorlist">true</attribute>
  </task>
  <task name="jsc">
   <attribute name="supportsplatform">true</attribute>
  </task>
  <task name="vjc">
   <attribute name="supportsnowarnlist">true</attribute>
   <attribute name="supportskeycontainer">true</attribute>
   <attribute name="supportskeyfile">true</attribute>
   <attribute name="supportsdelaysign">true</attribute>
  </task>
  <task name="resgen">
   <attribute name="supportsassemblyreferences">true</attribute>
   <attribute name="supportsexternalfilereferences">true</attribute>
  </task>
  <task name="delay-sign">
   <attribute name="exename">sn</attribute>
  </task>
  <task name="license">
   <attribute name="exename">lc</attribute>
   <attribute name="supportsassemblyreferences">true</attribute>
  </task>
 </tasks>
</framework>
```

NAnt Conventions

Build File Format

In order to promote consitency and readability, the .build files will be laid out in the following way. <TODO JRock: add the rest of the content>

Properties

Prefixes

Since properties defined in a .build file are visible to other .build files that use <include>, each project's .build file should use the file name for its prefix on properties. For example, the candle.build project should use the prefix 'candle.' before any local properties. Global properties (those that appear in the wix.include file) do not have a prefix.

Developing for Votive

If you want to contribute code to the Votive project or debug Votive, you must download and install the Visual Studio 2005 SDK, available at the <u>Visual Studio Extensibility Developer Center</u>. The Visual Studio 2005 SDK is non-invasive and will create an experimental hive in the registry that will leave your retail version of Visual Studio 2005 unaffected.

To start debugging Votive, set your breakpoints then press F5 in the Wix.sln for Visual Studio. The custom build actions in the Votive project will set up and register Votive in the experimental hive, so running Wix3.msi is not required, nor suggested.

Adding to the WiX Documentation

WiX documentation is compiled into the file WiX.chm as a part of the WiX build process. The source files for help are located in the wix\src\chm directory.

What the WiX help compiler does

The WiX help compiler does the following:

- Parses the file TOC.xml to determine the table of contents to construct in the CHM file and determine what HTML files to include in the CHM file.
- Includes all the .htm files listed in the project file in the list of documentation to build into the CHM
- Parses .xsd schema files referenced in TOC.xml and generates help topics for the attributes and elements that are annotated in the .xsd files.

How to add a new topic to WiX.chm

Adding a new topic to WiX.chm requires the following steps:

- Add a new HTML file with the contents of the new topic to the WiX source tree under src\chm\html.
- Add an entry for the new HTML to the src\chm\chm.proj file.
- Add any relevant images to the src\chm\imgs\ sub-directory in the WiX source tree.
- Add an entry for the new images to the src\chm\chm.proj file.
- Add a reference to the new HTML file to TOC.xml in the desired location in the table of contents.

Help topics may contain links to external Web pages, and may also contain relative links to other help topics or attributes or elements defined in one of the .xsd schema files.

To build the new content type *msbuild* from the command line in the src\chm directory.

Testing WiX

This section contains documents on how to create and execute tests for the Windows Installer XML Toolset.

- Running Tests
- Writing Tests

Running WiX Tests

There is a suite of tests that are included with WiX. They can be used to verify that changes to the toolset do not regress existing functionality.

Building the Tests

The tests will build as part of the normal WiX build. They have a dependency on Microsoft.VisualStudio.QualityTools.UnitTestFramework 9.0.0.0 assembly that ships with the following editions of Visual Studio:

- Visual Studio 2008 Professional Edition
- Visual Studio Team System 2008 Database Edition
- Visual Studio Team System 2008 Development Edition
- Visual Studio Team System 2008 Team Suite
- Visual Studio Team System 2008 Test Edition

The build system searches the registry to detect if one of the above mentioned editions is installed on the machine. If the <u>detection key</u> cannot be found then the tests will not build from Nant but they can still be built by MSBuild if the required UnitTestFramework assembly exists.

The tests are built into an assembly called wixtests.dll to the same location as the other WiX binaries.

Building the tests using Nant

Nant must be run from the WiX root directory. To build only the tests, specify the 'wixtests' target.

c:\delivery\dev\wix>nant.exe wixtests

Building the tests in Visual Studio

Open c:\delivery\dev\wix\test\wixtests.sIn from a WiX command window. The solution should build from within Visual Studio.

devenv.exe c:\delivery\dev\wix\test\wixtests.sln

Running the tests

The tests can be run from within Visual Studio or from the command line. Before the tests are run, the environment variable 'WIX_ROOT' must be set to the WiX root directory. It should be set if you are in a WiX command window, but if it is not:

set WIX_ROOT=c:\delivery\dev\wix

The WIX_ROOT environment variable requirement is used in many tests to locate test data.

Running the tests from the command line with MSTest.bat

There is a batch file, test.bat, which can be used to run the tests.

c:\delivery\dev\wix\test.bat [all|smoke|test name]

Running the tests from the command line with MSTest.exe

Run MSTest with the test binaries.

Running the tests from Visual Studio

Open wixtests.sln from a WiX command window.

devenv.exe c:\delivery\dev\wix\test\wixtests.sln

Run the tests from Visual Studio Test Manager.

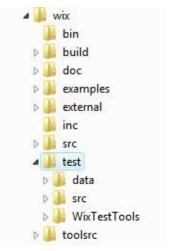
Writing WiX Tests

This document describes how to write tests for WiX.

Location of the Tests

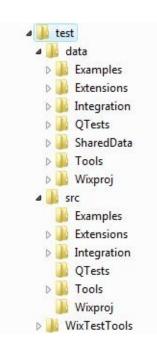
The root directory for the tests is %WIX_ROOT%\test. There are three main subdirectories:

- data: contains test data, eg wxs files
- src: contains source code for the tests
- WixTestTools: contains source code for the WixTestTools library



The *data* and *src* directories are further organized by feature area:

- Examples: Example tests
- Extensions: Tests for WiX extensions
- Integration: Tests for integration of two or more tools. Eg. Building an MSI from source with Candle and Light.
- QTests: Tests migrated from the previous test infrastructure
- SharedData: Test data that is shared across multiple tests
- Tools: Tests for a particular tool's command line options
- Wixproj: Tests for building .wixproj's with MSBuild



WixTests Solution

The test solution file, WixTests.sln, is located in %WIX_ROOT%\test\WixTests.sln. The WixTests solution currently contains two projects:

- WixTests: Contains all of the tests
- WixTestsTools: A library of wrapper classes and verification methods used by the tests

The solution should be opened from the WiX command window to ensure that the %WIX_ROOT% environment variable is set.

Example Tests

Example: Build and Verify an MSI

The following example shows how to test building an MSI from WiX source.

```
[TestMethod]
[Description("An example test that verifies an MSI is built correctly")]
[Priority(3)]
public void ExampleTest1()
{
    // Use the BuildPackage method to build an MSI from source
    string actualMSI =
Builder.BuildPackage(@"%WIX_ROOT%\test\data\SharedData\Authoring

    // The expected MSI to compare against
    string expectedMSI =
    @"%WIX_ROOT%\test\data\SharedData\Baselines\MSIs\BasicProduct.m

    // Use the VerifyResults method to compare the actual and expected
MSIs
    Verifier.VerifyResults(expectedMSI, actualMSI);
}
```

Example: Check for a Warning and Query an MSI

The following example shows how to build an MSI using the Candle and Light wrapper classes. It also demonstrates how to check for a warning from Light and guery the resuling MSI.

```
[TestMethod]
[Description("An example test that checks for a Light warning and queries the resulting MSI")]
[Priority(3)]
public void ExampleTest2()
```

```
// Compile a wxs file
  Candle candle = new Candle();
candle.SourceFiles.Add(@"%WIX ROOT%\test\data\Examples\Example
  candle.Run();
  // Create a Light object that uses some properties of the Candle object
  Light light = new Light(candle);
  // Define the Light warning that we expect to see
  WixMessage LGHT1079 = new WixMessage(1079,
WixMessage.MessageTypeEnum.Warning);
  light.ExpectedWixMessages.Add(LGHT1079);
  // Link
  light.Run();
  // Query the resulting MSI for verification
  string query = "SELECT `Value` FROM `Property` WHERE `Property`
= 'Manufacturer'":
  Verifier. VerifyQuery(light.OutputFile, query, "Microsoft Corporation");
}
```

Example: ICE Validation with Smoke

The following example shows how to verify that Smoke catches a particular ICE violation and how to use the Result object to perform further verification.

```
[TestMethod]
[Description("An example test that verifies an ICE violation is caught by smoke")]
[Priority(3)]
public void ExampleTest3()
{
    string testDirectory =
Environment.ExpandEnvironmentVariables(@"%WIX_ROOT%\test\data\E
```

```
// Build the MSI that will be run against Smoke. Pass the -sval
argument to delay validation until Smoke is run
  string msi = Builder.BuildPackage(testDirectory, "product.wxs",
"product.msi", null, "-sval");
  // Create a new Smoke object
  Smoke smoke = new Smoke();
  smoke.DatabaseFiles.Add(msi);
smoke.CubFiles.Add(@"%WIX ROOT%\test\data\Examples\ExampleTes
  // Define the expected ICE error
  WixMessage LGHT1076 = new WixMessage(1076, "ICE1000:
Component 'ExtraICE.0.ProductComponent' installs into directory
'TARGETDIR', which will get installed into the volume with the most free
space unless explicitly set.", WixMessage.MessageTypeEnum.Warning);
  smoke.ExpectedWixMessages.Add(LGHT1076);
  // Run Smoke and keep a reference to the Result object that is
returned by the Run() method
  Result result = smoke.Run();
  // Use the Result object to verify the exit code
  // Note: checking for an exit code of 0 is done implicitly in the Run()
method but
  // this is just for demonstration purposes.
  Assert.AreEqual(0, result.ExitCode, "Actual exit code did not match
expected exit code");
```

Developing WiX Extensions

This section covers the following topics for developers who want to create their own WiX extensions:

- Introduction to Developing WiX Extensions
- Creating a Skeleton Extension
- Creating a Preprocessor Extension

Introduction to Developing WiX Extensions

Common Requirements

In order to understand how each of the classes of extensions work, one should start by looking at the WiX source code. All extensions have the following things in common:

- Implemented using the .NET Framework 2.0. The rest of the WiX toolset currently only depends on the .NET Framework 2.0, so in order to ensure backwards compatibility, it is a best practice to develop new extensions so that they only depend on the .NET Framework 2.0 as well.
- Build a subclass of the appropriate extension object, which gives it an 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 extensible methods and virtual members which will get invoked at the approriate location during the single pass compile.
- Compiled into a DLL.
- Placed next to WiX EXEs along with all other WiX extension DLLs.
- Registered with WiX by passing the path of the exension DLL as a command line argument to the compiler and/or linker.

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 in which a particular class of extensions will be processed. As a result, there must not be any sequencing dependencies between extensions within the same extension class.

Extension developers might also implement a RequiredVersion attribute on the <u>Wix</u> element. This allows setup developers using your extension to require a specific version of the extension in case a new feature is introduced or a breaking change is made. You can add an attribute to the Wix element in an extension as shown in the following example.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:xse="http://schemas.microsoft.com/wix/2005/XmlSchemaExtension">
   <xs:attribute name="RequiredVersion" type="xs:string">
     <xs:annotation>
        <xs:documentation>
        The version of this extension required to compile the defining source.
        </xs:documentation>
        <xs:appinfo>
        <xs:appinfo>
        <xse:parent namespace="http://schemas.microsoft.com/wix/2006/wi" ref="
        </xs:appinfo>
        </xs:annotation>
        </xs:attribute>
        </xs:schema>
```

Extensions

WiX supports the following classes of extensions:

- **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 is written to binary form.
- **Binder Extensions** allow clients to modify the behavior of the Binder.
- **Decompiler Extensions** allow clients to decompile custom tables into XML.
- **Validator Extensions** allow clients to process validation messages. By default, validation messages are output to the console.
- **Mutator Extensions** allow clients to modify the behavior of the Mutator.
- **Harvester Extensions** allow clients to modify the behavior of the Harvester.
- **Unbinder Extensions** allow clients to modify the behavior of the Unbinder.

For information on how to use WiX extensions on the command line or inside the Visual Studio IDE, please visit the <u>Using WiX extensions</u> topic.

For information on how to use localized WiX extensions, please visit the <u>Localized extensions</u> topic.

Creating a Skeleton WiX Extension

WiX extensions are used to extend and customize what WiX builds and how it builds it.

The first step in creating a WiX extension is to create a class that extends the WixExtension class. This class will be the container for all the extensions you plan on implementing. This can be done by using the following steps:

- 1. In Visual Studio, create a new C# library (.dll) project named SampleWixExtension.
- 2. Add a reference to wix.dll to your project.
- 3. Add a using statement that refers to the Microsoft. Tools. Windows Installer Xml names pace.

using Microsoft.Tools.WindowsInstallerXml;

4. Make your SampleWixExtension class inherit from WixExtension.

public class SampleWixExtension : WixExtension {}

5. Add the AssemblyDefaultWixExtensionAttribute to your AssemblyInfo.cs.

[assembly: AssemblyDefaultWixExtension(typeof(SampleWixExtension.Sa

6. Build the project.

Although this WiX extension will not do anything yet, you can now pass the newly built SampleWixExtension.dll on the command line to the Candle and Light by using the -ext flag like the following:

candle.exe Product.wxs -ext SampleWixExtension.dll
light.exe Product.wxs -ext SampleWixExtension.dll

This covers the basics of creating the skeleton of an extension. You can

now use this skeleton code to build your own custom action. After you are done, you can author the custom action in the WiX source code by following the <u>Adding a Custom Action</u> topic. You can also build your own extensions to the WiX toolset using this skeleton code. For an example of building an extension, see <u>Creating a Preprocessor Extension</u>.

Creating a Preprocessor Extension

The preprocessor in WiX allows extensibilty at a few levels. This sample will demonstrate how to add a PreprocessorExtension to your WixExtension that will handle variables and functions you define in your own namespace.

This sample assumes you have already reviewed the <u>Creating a Skeleton Extension</u> topic.

- Add a new class to your project called SamplePreprocessorExtension.
- 2. If you added a new file for this class, add a using statement that refers to the Microsoft. Tools. Windows Installer Xml names pace.

```
using Microsoft.Tools.WindowsInstallerXml;
```

3. Make your SamplePreprocessorExtension class implement PreprocessorExtension.

```
public\ class\ Sample Preprocessor Extension: Preprocessor Extension
```

 Add your SamplePreprocessorExtension to your <u>previously created</u> <u>SampleWixExtension class</u> and override the PreprocessorExtension property from the base class. This will cause your extension to know what to do when WiX asks your extension for its preprocessor extension.

```
}
return this.preprocessorExtension;
}
```

5. In your SamplePreprocessorExtension class, specify the prefixes or namespaces that your extension will handle. For example, if you want to be able to define a variable named \$(sample.ReplaceMe), then you need to specify that your extension will handle the "sample" prefix.

```
private static string[] prefixes = { "sample" };
public override string[] Prefixes { get { return prefixes; } }
```

6. Now that you have specified the prefixes that your extension will handle, you need to handle variables and functions that are passed to you from WiX. You do this by overriding the GetVariable and EvaluateFunction methods from the PreprocessorExtension base class.

```
}
public override string EvaluateFunction(string prefix, string function, string
  string result = null;
  switch (prefix)
     case "sample":
       switch (function)
          case "ToUpper":
             if (0 < args.Length)
               result = args[0].ToUpper();
             else
               result = String.Empty;
             break;
       break;
  return result;
```

7. Build the project.

You can now pass your extension on the command line to Candle and expect variables and functions in your namespace to be passed to your extension and be evaluated. To demonstrate this, try adding the following properties to your WiX source file:

```
<Property Id="VARIABLETEST" Value="$(sample.ReplaceMe)" />
<Property Id="FUNCTIONTEST" Value="$(sample.ToUpper(uppercase))" />
```

The resulting .msi file will have entries in the Property table with the values "replaced" and "UPPERCASE" in the Property table.

Adding a Custom Action

This example shows how to author a binary custom action called "FooAction". A common example is a dll custom action that launches notepad.exe or some other application as part of their install. Before you start, you will need a sample dll that has an entrypoint called "FooEntryPoint". This sample assumes you have already reviewed the Creating a Skeleton Extension topic.

Step 1: Create a Fragment

You could directly reference the custom action in the same source file as the product definition. However, that will not enable the same custom action to be used elsewhere. So rather than putting the custom action definition in the same source file, let's exercise a little modularity and create a new source file to define the custom action called "ca.wxs".

Okay, that's it. We're done with editing the "ca.wxs" source file. That little bit of code should compile but it will not link. Remember linking requires that you have an entry section. A <Fragment/> alone is not an entry section. Go to the next step to link the source file.

Step 2: Add the custom action

We would need to link this source file along with a source file that contained <Product/> or <Module/> to successfully complete.

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
 <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'</p>
       Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
   <Package Description='My first Windows Installer package'</p>
       Comments='This is my first attempt at creating a Windows Installer datal
       Manufacturer='Microsoft Corporation' Installer Version='200' Compressed
   <Media Id='1' Cabinet='product.cab' EmbedCab='yes' />
   <Directory Id='TARGETDIR' Name='SourceDir'>
     <Directory Id='ProgramFilesFolder' Name='PFiles'>
       <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' Source='readme.txt'
        </Component>
        <Merge Id='MyModule' Language='1033' SourceFile='module.msm' D
       </Directory>
     </Directory>
   </Directory>
   <Feature Id='MyFeature' Title='My 1st Feature' Level='1'>
     <ComponentRef Id='MyComponent' />
     <MergeRef Id='MyModule' />
   </Feature>
   <InstallExecuteSequence>
     <Custom Action='FooAction' After='InstallFiles'/>
   </InstallExecuteSequence>
 </Product>
```

</Wix>

Those three lines are all you need to add to your Windows Installer package source file to call the "FooAction" 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 ca.wxs

C:\test> light product.wixobj ca.wixobj -out product.msi

C:\test> msiexec /i product.msi

Now as part of your installation, whatever "FooAction" is supposed to perform, you should see happen after the InstallFiles action.

Localizing Extensions

You can create your own localized extensions like <u>WixUIExtension</u> using lit.exe. Localized extensions can even contain multiple languages. Products using these extensions can pass the -cultures switch to <u>light.exe</u> along with the -ext switch to reference the extension.

WiX extensions contain libraries comprised of fragments. These fragments may contain properties, search properties, dialogs, and more. Just like when localizing products, replace any localizable fields with variables in the format !(loc.variableName). Product would be authored to reference elements in this library, and when compiled would themselves contain the localization variables. The following shows an example on how to localize an extension

Step 1: Author a WiX Fragment

Create a .wxs file named example.wxs and add the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<Fragment>
     <Error Id="50000">!(loc.errormsg)</Error>
</Fragment>
```

You have just authored a Fragment that will be compiled into a WiX library. It contains an error message that references a localized string.

Step 2: Author the Localization File

The WiX localization files, or .wxl files, are a collection of strings. For libraries, extension developers can choose whether or not those strings can be overwritten by .wxl files specified during linkage of the product. Create a .wxl file named en-us.wxl and add the following content:

```
<?xml version="1.0" encoding="utf-8"?>
<WixLocalization Culture="en-us" xmlns="http://schemas.microsoft.com/wix/20)
<String Id="errormsg" Overridable="yes">General Failure</String>
</WixLocalization>
```

These <u>String</u> elements are attributed as @Overridable="yes" to allow for product developers to override these strings with their own values if they so choose. For example, a product developer may wish to use "Previous" instead of "Back", so they can define the same String/@Id in their own .wxl while still linking to the extension where that string is used. This offers product developers the benefits of the library while allowing for customizations. Extension developers can also choose to disallow overriding certain strings if it makes sense to do so.

Step 3: Build the library

When both the fragment authoring and localization file are complete, they can be compiled and linked together using candle.exe and lit.exe.

First compile the .wxs source.

candle.exe example.wxs -out example1.wixobj

Now link together the .wixobj file and the .wxl file in the extension library.

lit.exe example.wixobj -loc en-us.wxl -out example.wixlib

You can add more than one .wxl file for each culture you want available. To be useful, the .wixlib should be embedded into a managed assembly and returned by WixExtension.GetLibrary().

Using the Libraries

Product developers reference elements within your .wixlib, as shown in the <u>WixUIExtension</u> example. When compiling and linking, the extension is specified on the command line using the -ext switch. If any additional localization variables are used in the product authoring or would override localization variables in the library, those .wxl files are passed to the -loc switch as shown in the example below.

candle.exe example.wxs -ext WixUIExtension -out example.wixobj light.exe example.wixobj -ext WixUIExtension -cultures:en-us -loc en-us.wxl -o

Unit-testing custom actions with Lux

Custom actions are a frequent cause of installation failures so it's important to test them thoroughly. Custom actions themselves usually aren't tested. The traditional testing approach is to run functional tests on an entire installer and to cover as many scenarios and platform combinations as possible.

Custom action patterns

WiX compiler extensions provide one way of improving custom action quality: Because compiler extensions run at build time instead of install time, they can perform all sorts of data validation and conversion on strongly-typed authoring before converting it to rows and columns of custom tables in the MSI package.

Immediate custom actions then read those custom tables, check current state (for example, component action state, the state of the machine itself), and serialize the resulting data into a custom action data property. Immediate custom actions are the place to do the logic that needs live state and cannot be determined at build time by a compiler extension. Because immediate custom actions run in the security context of the installing user and outside an installation transaction, they generally do not have permissions to modify the machine and if they fail, the installation simply ends without the need to do any cleanup or rollback.

Deferred custom actions read the custom action data property set by immediate custom actions to know what to do. One way to improve custom action reliability is to make as few decisions as possible in deferred custom actions; instead, implement all the logic in compiler extensions and immediate custom actions and have deferred custom actions simply read the custom action data property in a loop to modify the machine.

The WiX custom actions that modify the machine use this pattern. For example, XmlConfig authoring is validated by the WixUtilExtension compiler extension and translated to rows and columns in the XmlConfig table. The SchedXmlConfig immediate custom action reads the XmlConfig table, constructs a custom action data property based on the XmlConfig table and machine's state (including checking component state and storing existing file data to support rollback), then schedules the ExecXmlConfig deferred custom action to execute the XML changes and the ExecXmlConfigRollback rollback custom action to roll back the changes.

Testing with Lux

Lux is a WiX extension (and associated tools) that let you write datadriven unit tests for your custom actions. The executive summary: Lux runs your immediate custom actions then validates they set properties to the values you expect.

While it's a simple approach, if your custom actions are factored as discussed above, validating the properties set by immediate custom actions can validate all the interaction between your custom actions, the MSI package, and MSI itself.

If your custom actions aren't factored as discussed--for example, if your deferred custom actions expect only an installation directory and have logic to construct file paths from it--then it's likely that your immediate custom actions don't have a lot of logic that's useful to test.

Lux does not help you test the custom action code that actually modifies the machine; for that, continue to use other unit-test frameworks and automated tests. By working only with immediate custom actions, Lux can let MSI run the custom actions as-is, eliminating the need to write custom test doubles for the MSI API. Lux runs from a per-user package so unless you run the tests from an elevated command prompt, none of the custom actions get elevated privileges and therefore cannot modify the machine.

Here's how Lux works:

- 1. You write your unit tests using XML in WiX source files.
- 2. The Lux extension converts the XML to a table in a test .msi package.
- 3. The Lux custom action runs after all other immediate custom actions and evaluates your unit tests.

Authoring unit tests

Lux supports the following unit tests:

- Property values
- Expressions
- Multi-value properties
- Name/value-pair properties

Note that you should always author unit tests in fragments separate from your custom action authoring or any other product authoring. If you mix unit tests with other authoring, WiX includes the unit-test data in your "real" installers.

Property value tests

A simple test lets you specify a property to test, a value to test against, and the operator to compare with (which defaults to "equal").

```
<Fragment>
  <lux:UnitTest CustomAction="TestCustomActionSimple" Property="SIMPLE"
  </Fragment>
```

When the test runs, Lux compares the value of the SIMPLE property against the (formatted) value [INSTALLLOCATION]. If the two match (because the operator is "equal"), the test passes. Legal values of the Operator attribute are:

equal

(Default) Compares Property to Value and succeeds if they are equal.

notEqual

Compares Property to Value and succeeds if they are NOT equal.

caseInsensitiveEqual

Compares Property to Value and succeeds if they are equal (ignoring case).

caseInsensitiveNotEqual

Compares Property to Value and succeeds if they are NOT equal (ignoring case).

Test conditions

Conditions let you validate code paths in your custom action. For example, if your custom action behaves differently on Windows XP than it does on Windows Vista and later, you can create two tests with mutually exclusive conditions:

If a test has a condition, the test runs only if its condition is true.

Expression tests

Expression tests let you test any valid MSI expression. If the expression is true, the test passes. If the expression is false or invalid, the test fails.

```
<Fragment>
  <lux:UnitTest CustomAction="TestCustomActionSimple">
        <lux:Expression>NOT MsiSystemRebootPending AND SIMPLE</lux:Expression>UnitTest>

<
```

Multi-value property tests

Because deferred custom actions can access only a single custom-action data property, custom actions that need more than one piece of data encode it in a single string. One way is to have the immediate custom action separate multiple elements with a known separator character, then have the deferred custom action split the string at those separate characters. Lux supports such separators using the ValueSeparator and Index attributes.

A condition under the parent UnitTest element applies to all individual unit tests. Override it with a Condition child element.

Name/value-pair property tests

Another way of providing multiple values to a deferred custom action is to combine name/value pairs into a single string. Lux supports name/value-pair properties using the NameValueSeparator and Index attributes.

Test mutations

Immediate custom actions frequently need to create different custom action data depending on global machine state. For example, if a component is already installed, a custom action might have different behavior to upgrade the component, versus installing it for the first time.

Because Lux runs only immediate custom actions, it's not possible to actually update the global machine state. One approach is to create multiple custom action DLLs, mocking MSI functions to return hard-coded values. Lux simplifies this model with *test mutations*.

Test mutations let you author unit tests with different expected results. The mutation id is passed as the value of the WIXLUX_RUNNING_MUTATION property. Your custom action, typically in an '#ifdef DEBUG' block, retrieves the WIXLUX_RUNNING_MUTATION property and mock different behavior based on the mutation. To author test mutations, use the Mutation element with UnitTest elements as children. For example:

```
<lux:Mutation Id="SimulateDiskFull">
  <lux:UnitTest ... />
  </lux:Mutation>
```

Nit runs the test package once for each mutation, setting the WIXLUX_RUNNING_MUTATION property to one mutation id at a time. Tests that aren't children of a mutation are run every time.

Building test packages

Lux unit tests run from a minimal package that includes just your unit tests and the resources they need to run. Because Lux runs only immediate custom actions, it doesn't need a full, per-machine package that includes all the files and other resources to be installed. Such a minimal package saves build time but does require that your WiX source code be well modularized with fragments. For example, you should always author unit tests in fragments separate from any other authoring. If you mix unit tests with other authoring, WiX includes the unit-test data in your "real" installers. Likewise, any other WiX authoring included in unit-test fragments is included in test packages.

Lux comes with a tool that simplifies the creation of test packages. Its name is lux.exe. To use lux.exe:

- 1. Compile the source file containing your unit tests.
- 2. Run lux.exe on the .wixobj file and specify a source file for the test package.
- 3. Compile the test package source.
- 4. Link the test package .wixobj with the unit tests .wixobj.

For example:

candle -ext WixLuxExtension CustomActions.wxs
lux CustomActions.wixobj -out LuxSample1_test.wxs
candle -ext WixLuxExtension LuxSample1_test.wxs
light -ext WixLuxExtension LuxSample1_test.wixobj CustomActions.wixobj -ou

Lux also includes an MSBuild task and .targets file to let you build test packages from the same .wixproj you use to build your installers. To build a test package, build the BuildTestPackage target using MSBuild 3.5:

%WINDIR%\Microsoft.NET\Framework\v3.5\MSBuild.exe /t:BuildTestPackag

Running unit tests

After building the test package, you can run it with logging enabled to capture test results:

msiexec /l test1.log /i bin\Debug\LuxSample1_test.msi

Search the log for **WixRunImmediateUnitTests** to see test results and other logging from the Lux custom action.

Nit: The Lux test runner

Lux also includes Nit, a console program that monitors the logging messages emitted by unit tests and reports success or failure. To use Nit on your test packages, just specify their filenames as arguments to nit.exe. For example:

nit LuxSample1_test.msi

Lux also lets you run Nit on your test packages from the same .wixproj you use to build your installers. To run a test package under Nit, build the Test target using MSBuild 3.5:

%WINDIR%\Microsoft.NET\Framework\v3.5\MSBuild.exe /t:Test

The test package will be built before the tests are run, if necessary. The output looks like the following, with failing tests highlighted in red as build errors:

Test:

Microsoft (R) Windows Installer Xml Unit Test Runner version 3.5.1204.0 Copyright (C) Microsoft Corporation. All rights reserved.

Test luxB21F0D12E0701DBA30FFB92A532A5390 passed: Property 'SIMPLE' Test TestConditionBeforeVista passed: Property 'SIMPLE' matched expected variet TestConditionVistaOrLater passed: Property 'SIMPLE' matched expected variet TestExpressionTruth passed: Expression 'NOT MsiSystemRebootPending nit.exe: error NIT8103: Test luxA6D27EC5903612D7F3786FF71952E314 faile Test lux210257649C16AFA33793F1CDDF575505 passed: Property 'MULTIV nit.exe: error NIT8103: Test lux402940A90D3ADAD181D599AB8C260FA0 far Test lux453EC8DB458A8F66F0D22970CFF2AE99 passed: Property 'NAMEN Test lux20CB4F88795F22D15631FD60BA03AFEB passed: Property 'NAMEN nit.exe: error NIT8102: 2 tests failed. 7 tests passed.

Done Building Project "C:\Delivery\Dev\wix35\src\lux\samples\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxSample1\LuxS

Build FAILED.

"C:\Delivery\Dev\wix35\src\lux\samples\LuxSample1\LuxSample1.wixproj" (Te (Test target) ->

nit.exe: error NIT8103: Test luxA6D27EC5903612D7F3786FF71952E314 fai nit.exe: error NIT8103: Test lux402940A90D3ADAD181D599AB8C260FA0

nit.exe: error NIT8102: 2 tests failed. 7 tests passed.

0 Warning(s)
3 Error(s)

Time Elapsed 00:00:07.87

FAQ

Are these really unit tests? They look a lot like **<u>Fit tests</u>**.

Fit tests are tabular and data-driven, so they have a lot in common with Lux's unit tests. But fit tests are focused on high-level outputs, whereas unit tests are low-level developer tests.

Using the custom action code as-is sounds good, but are there any limitations with that approach?

Yes. Because you are running the actual custom action, any code paths that rely on machine state reflect the state of the machine you run the tests on. For example, code that has different behavior on different versions of Windows runs only one way, just like it does in a normal installer. You can add debug code that looks for the presence of the WIXLUXTESTPACKAGE property; it's set to 1 in a test package.

I have unit tests that fail because directory properties are being returned as empty strings. Why?

The most likely cause is that your directories are defined as children of your installer's Product element. Lux.exe builds its own Product element to product a minimal test package, so none of the resources defined in your Product are available to the unit tests. The simplest solution is to move those resources to their own Fragment.

Do I have to write my custom actions in C++?

No, Lux works with any immediate custom actions, regardless of the language they're written in, including MSI type 51 property-setting custom actions.