

Windows Installer XML (WiX) v3.0 Help

This documentation contains information about the Windows Installer XML (WiX) toolset. See the following topics for more detailed information:

[Introduction](#)

[About WiX](#)

[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](#)

Introduction to Windows Installer XML (WiX) toolset

What is WiX?

WiX is a set of tools and specifications that allow you to easily create Windows Installer database files. The WiX tools model the traditional compile and link model used to create executables from source code. For WiX, source code is written in XML files. These files are validated against a schema, `wix.xsd`, then processed by a preprocessor, compiler, and linker to create the desired result. WiX has been designed to allow for the easy creation of multiple Windows Installer databases from a small set of source files. You can use WiX both on the command line and in the Visual Studio development environment.

Additional Resources

[Getting Started Learning WiX](#)

About WiX

Windows Installer XML, or WiX, provides a schema that describes a Windows Installer database (MSI or MSM), as well as tools to convert the XML description files into a usable database. The WiX tools model the traditional compile and link model used to create executables from source code. This section provides an introduction on the tools and concepts as well as an overview of the file types in WiX.

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

[File Types](#)

[Tools and Concepts](#)

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 output is in the form of standard Windows Installer database files.

Source files (.wxs and .wxi) are compiled, producing object files (.wixobj). These object 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	Type	Description
.wxi	WiX 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.
.wxl	WiX Localization File	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.
.wxs	WiX Source File	A .wxs file is analogous to a .cpp file for C++. The Root element of this file is <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.
.wixout	WiX XML 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	WiX Library 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.

.wixpdb	WiX Debug File	A .wixpdb file is created by the linker for each final output. It contains the debugging information.
.wixmsp	WiX XML Patch File	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. For more information on .msi files, see the Windows Installer documentation .
.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. For more information on .msm files, see the Windows Installer documentation .
.mst	Windows Installer Transform	A transform file (.mst) is used to apply changes to an .msi file. For more information on .mst files, see the Windows Installer documentation .
.pcp	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. For more information on .pcp files, see the Windows Installer documentation .

Additional Information

Windows Installer XML Files - .wxs & .wixobj

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

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 <table>, <row> and <field> elements that provide the raw data to be placed in the Windows Installer database. In many cases, the linker will not only process the symbols and references but also use and update the raw data from the .wixobj file. It is interesting to note that the object file schema, objects.xsd, uses camel casing where the source file schema, wix.xsd, uses Pascal casing. This was a conscious choice to indicate that the object files are not intended to be edited by the user. In fact, all schemas that define data to be processed only by the WiX tools use camel casing.

Tools and Concepts

This section explores the concepts of the preprocessor, compiler, and linker. In addition, it provides a complete list of tools that WiX offers.

[List of tools](#)

[Preprocessor](#)

[Compiler \(Candle\)](#)

[Linker \(Light\)](#)

[Library Tool \(Lit\)](#)

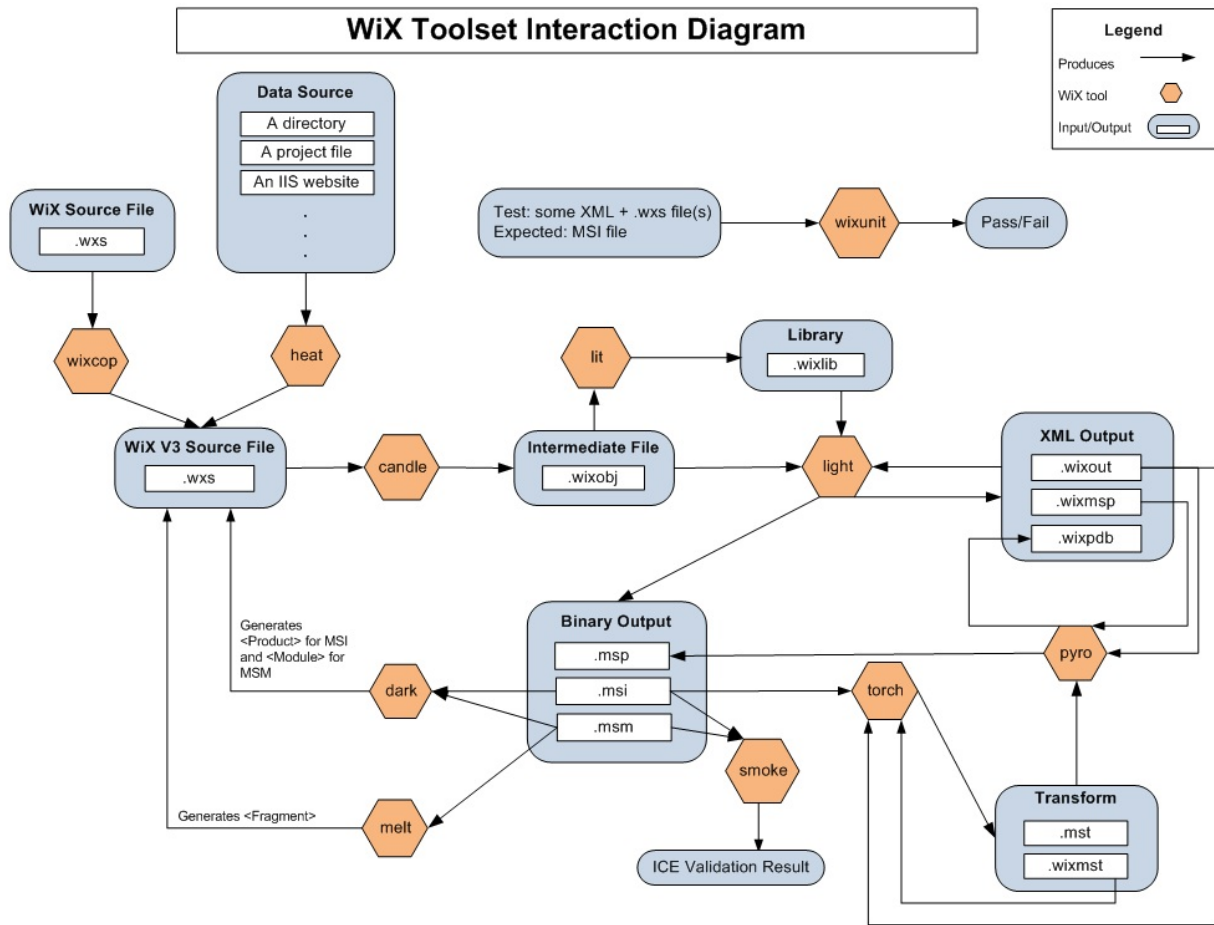
[Harvester \(Heat\)](#)

[Decompiler \(Dark\)](#)

[WixCop](#)

Diagram

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



Last Updated: 12/19/2007

List of Tools

Name	Description
Candle	Preprocesses and compiles WiX source files into object files (.wixobj).
Light	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.
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.
Heat	Generates WiX authoring from various input formats.
Melt	Converts an .msm into a component group in a WiX source file.
Torch	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.
Pyro	Takes an XML output patch file (.wixmsp) and one or more XML transform files (.wixmst) and produces an .msp file.
WixCop	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.

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.

Preprocessor

Introduction

Often you will need to add different pieces of your setup during build time depending on many factors such as the SKU being built. This is done by using conditional statements that will filter the xml before it is sent to the WiX compiler (candle). If the statement evaluates to true, the block of xml will be sent to candle. If the statement evaluates to false, candle will never see that section of xml.

The conditional statements are Boolean expressions based on environment variables, variables defined in the xml, literal values, and more.

Example

Let's start with an example. Say you want to include a file if you're building the "Enterprise SKU." Your build uses an environment variable %MySku%=Enterprise to specify this sku.

When you build the enterprise sku, this file will be included in the xml passed on to candle. When you build a different sku, the xml from EnterpriseFeature.wxs will be ignored.

```
<?if $(env.MySku) = Enterprise ?>  
  <?include EnterpriseFeature.wxs ?>  
<?endif ?>
```

Include Files <?include?>

As shown in the example above, files can be included by using the include tag. The filename referenced in the tag will be processed as if it were part of this file.

The root element of the include file must be <Include>. There are no other requirements beyond the expected wix schema. For example,

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


Variables

Any variable can be tested for its value or simply its existence. Custom variables can also be defined in your xml.

Three types of variables are supported:

\$(env._NtPostBld)

Gets the environment variable %_NtPostBld%

\$(sys.CURRENTDIR)

Gets the system variable for the current directory

\$(var.A)

Gets the variable A that was defined in this xml

The preprocessor evaluates variables throughout the entire document, including in <?if?> expressions and attribute values.

Environment Variables

Any environment variable can be referenced with the syntax \$(env.VarName). For example, if you want to retrieve the environment variable %_BuildArch%, you would use \$(env._BuildArch). Environment variable names are case-insensitive.

System Variables

WiX has some built-in variables. They are referenced with the syntax \$(sys.VARNAME) and are always in upper case.

CURRENTDIR - the current directory where the build process is running

SOURCEFILEPATH – the full path to the file being processed

SOURCEFILEDIR – the directory containing the file being processed

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 ?>

<?if $(var.Iexist)           ?><?endif?> <!-- true -->
<?if $(var.myValue) = 6     ?><?endif?> <!-- false -->
<?if $(var.myValue)!=3     ?><?endif?> <!-- false -->
<?if not "x"= "y"?>         <?endif?> <!-- true -->
```

```
<?if $(env.systemdrive)=a?><?endif?> <!-- false -->
<?if 3 < $(var.myValue)?> <?endif?> <!-- false -->
<?if $(var.B) = "good VAR"?> <?endif?> <!-- false -->
<?if $(var.A) and not $(env.MyEnvVariable) ?> <?endif?> <!--
<?if $(var.A) Or ($(var.B) And $(var.myValue) >=3)?><?endif?> <!--
<?ifdef IExist ?> <!-- true -->
  <?else?> <!-- false -->
<?endif?>
```

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:

```
<?foreach LCID in "1033;1041;1055"?>
    <Fragment Id='Fragment.$(var.LCID)'\>
        <DirectoryRef Id='TARGETDIR'\>
            <Component Id='MyComponent.$(var.LCID)'\ />
        </DirectoryRef>
    </Fragment>
<?endforeach?>
```

or

```
<?define LcidList=1033;1041;1055?>
<?foreach LCID in $(var.LcidList)?>
    <Fragment Id='Fragment.$(var.LCID)'\>
        <DirectoryRef Id='TARGETDIR'\>
            <Component Id='MyComponent.$(var.LCID)'\ />
        </DirectoryRef>
    </Fragment>
<?endforeach?>
```

or

```
filename: ExtentOfLocalization.wxi
<Include>
```

```
    <?define LcidList=1033;1041;1055?>
</Include>

and

<?include ExtentOfLocalization.wxi ?>
<?foreach LCID in $(var.LcidList)?>
    <Fragment Id='Fragment.$(var.LCID) '>
        <DirectoryRef Id='TARGETDIR'>
            <Component Id='MyComponent.$(var.LCID)' />
        </DirectoryRef>
    </Fragment>
<?endforeach?>
```

An alternative to the foreach process would be to write the template WiX fragment into a separate file and have another process generate the authoring that will be passed to WiX. The greatest merit of this alternative is that it's easier to debug.

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 [extensions](#) 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.

Usage Information

```
light.exe [-?] [-b basePath] [-nologo] [-out outputFile] objectFile
```

Light supports the following command line parameters:

Switch	Meaning
-ai	Allow identical rows; identical rows will be treated as a warning
-au	Allow unresolved references; this will cause invalid output to be created
-b	Specify a base path to locate all files; the default value is the current working directory
-bf	Bind files into a wixout; this switch is only valid when also providing the -xo option
-cc	Specify a path to cache built cabinet files; the path will not be deleted after linking
-ct <N>	Specify the number of threads to use when creating cabinets; the default is the %NUMBER_OF_PROCESSORS% environment variable
-cultures: <cultures>	Specifies a semicolon or comma delimited list of localized string cultures to load from .wxi files and libraries. Precedence of cultures is from left to right. For more information see Specifying cultures to build .
-cub	Provide a .cub file containing additional internal

	consistency evaluators (ICEs) to run
-d<name>= <value>	Define a WiX variable
-ext	Specify an extension assembly
-fv	Add a FileVersion attribute to each assembly in the MsiAssemblyName table (rarely needed)
-loc <loc.wxl>	Provide a .wxl file to read localization strings from
-nologo	Skip printing Light logo information
-notidy	Prevent Light from deleting temporary files after linking is complete (useful for debugging)
-out	Specify an output file; by default, Light will write to the current working directory
-pedantic	Display pedantic output messages
-reusecab	Reuse cabinets from the cabinet cache instead of rebuilding cabinets
-sa	Suppress assemblies: do not get assembly name information for assemblies
-sacl	Suppress resetting ACLs (useful when laying out an image to a network share)
-sadmin	Suppress adding default Admin sequence actions
-sadv	Suppress adding default Advt sequence actions

-sdut	Suppress dropping unreal tables to the output image; this switch is set by default when the -xo switch is provided
-sice:<ICE>	Suppress running internal consistency evaluators (ICEs) with specific IDs
-sma	Suppress processing the data in the MsiAssembly table
-sf	Suppress files: do not get any file information; this switch is equivalent to the combination of the -sa and -sh switches
-sh	Suppress file information: do not get hash, version, language and other file properties
-sl	Suppress layout creation
-ss	Suppress schema validation for documents; this switch provides a performance boost during linking
-sui	Suppress adding default UI sequence actions
-sv	Suppress intermediate file version mismatch checking
-sval	Suppress MSI/MSM validation
-sw<N>	Suppress warnings with specific message IDs
-ts	Tag sectionId attribute on rows; this switch is set by default when the -xo switch is provided
-tsa	Tag sectionId attribute on rows and generate the rows when null; this switch is set by default when the -xo switch

	is provided
-usf <output.xml>	Specify an unreferenced symbols file
-v	Generate verbose output
-wx	Treat warnings as errors
-xo	Generate XML output instead of an MSI
-?	Display Light help information

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
<code>bind.fileLanguage.<i>FileID</i></code>	<code>!(bind.fileLanguage.MyFile)</code>	1033
<code>bind.fileVersion.<i>FileID</i></code>	<code>!(bind.fileVersion.MyFile)</code>	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
<code>bind.assemblyCulture.<i>FileID</i></code> <i>(managed only)</i>	<code>!(bind.assemblyCulture.MyAs</code>
<code>bind.assemblyFileVersion.<i>FileID</i></code>	<code>!(bind.assemblyFileVersion.M</code>
<code>bind.assemblyFullName.<i>FileID</i></code>	<code>!(bind.assemblyName.MyAss</code>

<i>(managed only)</i>	
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> <i>(native only)</i>	!(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 diferent .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 [WixVariable](#) 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.

Library Tool (lit)

Lit is the WiX library creation tool. It can be used to combine multiple .wixobj files into libraries that can be consumed by [light](#).

Usage Information

```
lit.exe [-?] [-nologo] [-out libraryFile] objectFile [objectFile ..
```

Lit supports the following command line parameters:

Switch	Meaning
-b	Specify a base path to locate all files; the default value is the current working directory
-bf	Bind files into the library file
-ext	Specify an extension assembly
-loc <loc.wxl>	Provide a .wxl file to read localization strings from
-nologo	Skip printing Lit logo information
-out	Specify an output file; by default, Lit will write to the current working directory
-ss	Suppress schema validation for documents; this switch provides a performance boost during linking
-sv	Suppress intermediate file version mismatch checking
-sw<N>	Suppress warnings with specific message IDs
-v	Generate verbose output

-wx	Treat warnings as errors
-?	Display Lit help information

Dark

Dark is a tool for converting a Windows Installer database (.msi, .msm, .msp, .mst, .pcp) into a WiX source file. This tool is very useful for getting all your authoring into a WiX source file when you have an existing Windows Installer database. However, you will then need to tweak this file to accommodate different languages and breaking things into fragments.

Heat

Generates WiX authoring from various input formats.

Every time heat is run it regenerates the output file and any changes are lost.

Usage Information

```
heat.exe [-?] harvestType <harvester arguments> -out sourceFile.wxs
```

Heat supports the harvesting types:

Harvest Type	Meaning
dir	Harvest a directory.
file	Harvest a file.
project	Harvest outputs of a Visual Studio project.
website	Harvest an IIS web site.
perf	Harvest performance counters from a category.

Heat supports the following command line parameters:

Switch	Meaning
-ag	Auto generate component guids at compile time, e.g. set Guid="*".
-gg	Generate guids now. All components are given a guid when heat is run.
-g1	Generate component guids without curly braces.
-ke	Keep empty directories.

-nologo	Skip printing heat logo information.
-out	Specify output file (default: write to current directory).
-pog:<group>	<p>Specify output group of Visual Studio project, one of: Binaries, Symbols, Documents, Satellites, Sources, Content.</p> <ul style="list-style-type: none"> • Binaries - primary output of the project, e.g. the assembly exe or dll. • Symbols - debug symbol files, e.g. pdb. • Documents - documentation files. • Satellites - the localized resource assemblies. • Sources - source files. • Content - content files. <p>This option may be repeated for multiple output groups; e.g. -pog:Binaries -pog:Content.</p>
-scom	Suppress COM elements.
-sfrag	Suppress generation of fragments for directories and components.
-sreg	Suppress registry harvesting.
-suid	Suppress unique identifiers for files, components, & directories.
-sw<N>	Suppress all warnings or a specific message ID, e.g. -sw1011 -sw1012.
-swall	Suppress all warnings (<i>deprecated</i>).
-svb6	Suppress VB6 COM registration entries. When registering a COM component created in VB6 it adds

	<p>registry entries that are part of the VB6 runtime component, recommend for VB6 components to avoid breaking the VB6 runtime on uninstall.</p> <p>The following values are excluded:</p> <ul style="list-style-type: none"> - CLSID\{D5DE8D20-5BB8-11D1-A1E3-00A0C90F2731} - Typelib\{EA544A21-C82D-11D1-A3E4-00A0C90AEA82} - Typelib\{000204EF-0000-0000-C000-000000000046} - Any Interfaces that reference these two type libraries
-t:<xsl>	Transform harvested output with XSL file.
-indent <n>	Indentation multiple (overrides default of 4).
-template:<template>	Use template, one of: fragment, module, product. Default: fragment.
-v	Verbose output.
-wx[N]	Treat all warnings or a specific message ID as an error. e.g. -wx1011 -wx1012.
-wxall	Treat all warnings as errors (<i>deprecated</i>).
-? -help	Display heat help information.

Command line examples

Harvest a directory

```
heat dir -gg -sfrag -template:fragment -out directory.wxs ".\My Fil
```

This will harvest the sub folder "My Files" as a single fragment to the file directory.wxs. It will generate guids for all the files as they are found.

Harvest a file

```
heat file -ag -template:fragment -out file.wxs ".\My Files\File.dll
```

This will harvest the file "File.dll" as a single fragment to the file file.wxs. The component guid will be set to "*".

Harvest a Visual Studio project

```
heat project -pog:Binaries -ag -template:fragment -out project.wxs
```

This will harvest the binary output files from the Visual Studio project "MyProject.csproj" as a single fragment to the file project.wxs. The component guid will be set to "*".

Harvest a Website

```
heat website -template:fragment -out website.wxs "Default Web Site"
```

This will harvest the website "Default Web Site" as a single fragment to the file website.wxs.

Harvest a VB6 COM component

```
heat file -ag -template:fragment -svb6 -out vb6file.wxs ".\My Files
```

This will harvest the VB6 COM component "VB6File.dll" as a single fragment to the file vb6file.wxs and suppress the VB6 runtime specific registry entries.

Harvest performance counters

```
heat perf "My Category" -out perf.wxs
```

This will harvest all the performance counters from the category "My Category".

WixCop

WixCop is a WiX v3 command-line tool that 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.

The following table describes the switches that WixCop supports.

WixCop switch	Description
-?	Show help.
-nologo	Don't show the WixCop banner.
-f	Fix errors encountered in source files. This switch takes effect only for source files that are writable.
-s	Look for source files in subdirectories.
-indent: <i>n</i>	Overrides the default number of spaces per indentation level (4) to the number <i>n</i> you specify.
-set1 <i>filename</i>	Loads a primary settings file (see below). Note that there are no characters separating -set1 and the settings file name.
-set2 <i>filename</i>	Loads an alternate settings file that overrides some or all of the settings in the primary settings file. Note that there are no characters separating -set2 and the settings file

name.

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>
  <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 strongly 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.
ProgIdIconFormatted	Displayed when a ProgId/@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.

Using WiX on the command line

WiX can create Windows Installer databases which include: Windows Installer packages (.msi files), and merge modules (.msm files). We'll start by creating a Windows Installer package so that you'll have something that you can install and uninstall quickly. Then, we'll create a merge module and merge it into our example Windows Installer package.

[Authoring your first .wxs file](#)

[Creating Merge Modules](#)

[MSI Tables to WiX Schema](#)

Authoring your first .wxs file

The goal of this tutorial is to help you to get familiar with WiX and the fundamental 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.

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/2
</Wix>
```

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

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

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

C:\test>
```

Notice that when there is no error, Candle doesn't print any text other than its header. In fact, you can even suppress the header output by specifying "-nologo" on the command line. In that case, Candle will print nothing unless there is a failure.

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

```

<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
  <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'
    <Package Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Win
        Manufacturer='Microsoft Corporation' InstallerVersio

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Component Id='MyComponent' Guid='PUT-GUID-HERE' />
    </Directory>

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

```

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.

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

Note: This .msi requires administrative privileges in order to install correctly. It will silently fail if you are not installing as an Administrator.

```

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

product.wxs

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

C:\test> msiexec /i product.msi

```

You can now go to Add/Remove Programs in the Control Panel and see

"Test Package" listed there. At this point, you should go ahead and uninstall the package, so you don't forget it later.

Now that we have an empty .msi that installs and uninstalls properly, let's actually install something. So, 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/2
  <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'
    <Package Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Wi
        Manufacturer='Microsoft Corporation' InstallerVersi

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

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDir' Name='Test Program'>
          <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
            <File Id='readme' Name='readme.txt' DiskId='1' So
          </Component>
        </Directory>
      </Directory>
    </Directory>

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

Now you can compile, link, and install the .msi and 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 Merge Modules

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

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

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

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
  <Module Id='TestModule' Language='1033' Version='1.0.0.0'>
    <Package Id='PUT-GUID-HERE' Description='My first Merge Modul
      Comments='This is my first attempt at creating a Wi
      Manufacturer='Microsoft Corporation' InstallerVersi

  </Module>
</Wix>
```

You can, if you wish, compile and link that code. You will get a very small and not very interesting .msm file from the output of light.exe. So, let's add a text file to this Merge Module as we did to the Windows Installer package. First, create a text file called readme2.txt and put a different message than the message in readme1.txt to yourself in there. Then, update the source code to include the new file:

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
  <Module Id='TestModule' Language='1033' Version='1.0.0.0'>
    <Package Id='PUT-GUID-HERE' Description='My first Merge Modul
      Comments='This is my first attempt at creating a Wi
      Manufacturer='Microsoft Corporation' InstallerVersi

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='MyModuleDirectory' Name='.'>
        <Component Id='MyModuleComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme2' Name='readme2.txt' Source='readme
        </Component>
```

```
    </Directory>
  </Directory>
</Module>
</Wix>
```

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

Incorporating a Merge Module into a .wxs File

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

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
  <Product Id='PUT-GUID-HERE' Name='Test Package' Language='1033'
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'
    <Package Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Win
        Manufacturer='Microsoft Corporation' InstallerVersio

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

  <Directory Id='TARGETDIR' Name='SourceDir'>
    <Directory Id='ProgramFilesFolder' Name='PFiles'>
      <Directory Id='MyDir' Name='Test Program'>
        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
          <File Id='readme' Name='readme.txt' DiskId='1' So
        </Component>

        <Merge Id='MyModule' Language='1033' SourceFile='mod
      </Directory>
    </Directory>
  </Directory>

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

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

MSI Tables to WiX Schema

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

DuplicateFile Table

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

LaunchCondition Table

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

LockPermissions Table

This is authored using [Permission](#).

MoveFile Table

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

PublishComponent Table

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

MSI

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

WiX

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

RemoveIniFile

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

RemoveRegistry Table

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

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.

[WiX Project Types](#)

[WiX Item templates](#)

[WiX Project property pages](#)

[Creating a simple setup](#)

[Adding Project references](#)

[Adding WiX references](#)

Project Templates

Introduction

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

WiX Project

WiX Library Project

WiX Merge Module Project

WiX Project

A WiX project provides a starting point that can be 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

A WiX library project provides a starting point that can be 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

A WiX merge module project provides a starting point that can be 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.

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 [File List](#) 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

Introduction

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 temporary 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 pre-build 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 .wixlib's)

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

Creating a simple setup

To get started using WiX in Visual Studio, you can create a new WiX project in the Visual Studio IDE by using the following steps:

1. Click **File**, then click **New**, then click **Project...**
2. Choose the **WiX** node in the **Project types** tree, then select **WiX Project**, and name your project "MySetup" as depicted in Figure 1.

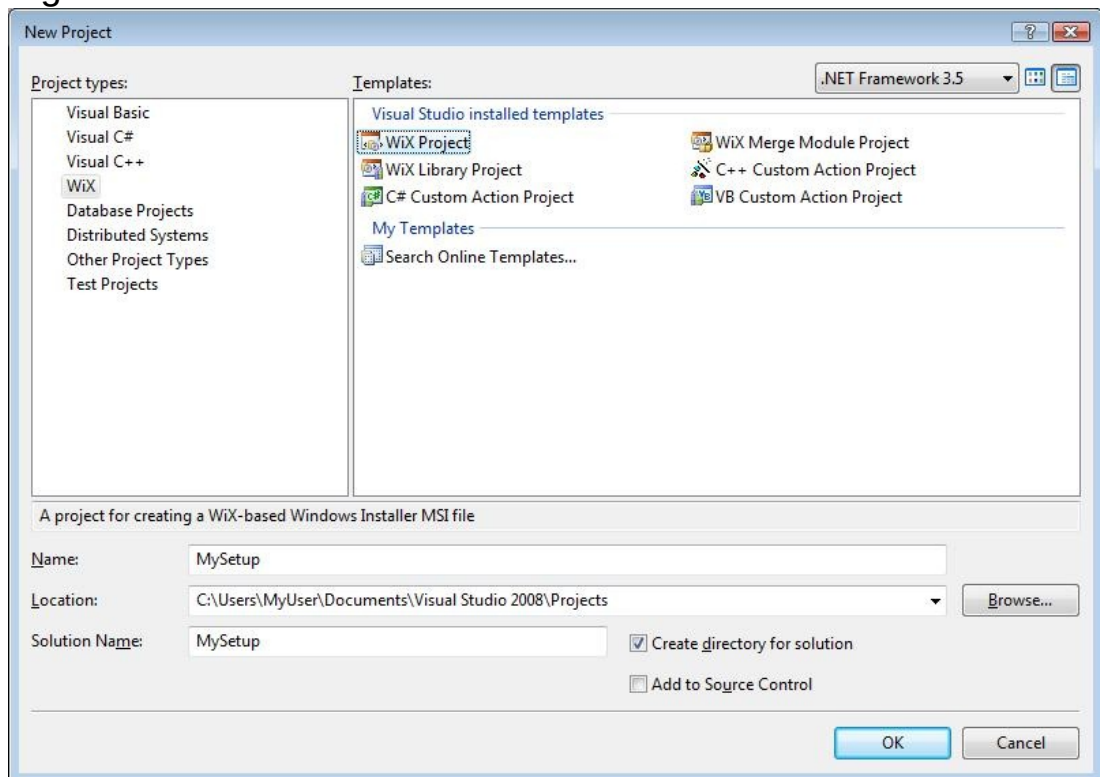


Figure 1. Create a new WiX project

This will create a new solution with the MySetup project and a MySetup.wxs file. The MySetup.wxs file requires some additional information before it will compile successfully. So let's take a moment to look at the MySetup.wxs file and discuss what needs to be done to it in order to be able to build an MSI.

A First Look at the Default MySetup.wxs File

The MySetup.wxs file contains the beginning of the setup code for the project. Everything needed to create an MSI can be added to this file. To begin with, let's look at the default contents provided in MySetup.wxs.

Note: If you are not familiar with Windows Installer setup packages, you are strongly encouraged to review the MSDN documentation about the [Installation Package](#) 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="MySe
    Version="1.0.0.0" Manufacturer="MySetup" UpgradeCode

  <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 Componen
        <!-- <Component Id="ProductComponent" Guid="78576f7
          <!-- TODO: Insert files, registry keys, and
          <!-- </Component> -->
        </Directory>
      </Directory>
    </Directory>
  </Directory>

  <Feature Id="ProductFeature" Title="PUT-FEATURE-TITLE-HERE"
    <ComponentRef Id="ProductComponent" />
  </Feature>
</Product>
</Wix>
```

If you are familiar with the Windows Installer, the structure of the MySetup.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 auto-complete in Visual Studio via IntelliSense. Next, the Product element defines the required Windows Installer properties used to identify the product, such as the [ProductCode](#), [ProductName](#), [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](#), [Component table](#), and [Feature table](#). The ComponentRef element is used to tie the Features to their Components which maps to the entries in the [FeatureComponents table](#).

Building the MySetup.wxs File

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

The cabinet 'InstallPackage.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. Nevertheless, an installation package named InstallPackage.msi was built in the bin/Debug folder, together with a file that is named InstallPackage.wixpdb, which contains debugging information. Running the InstallPackage.msi at this point does almost nothing.

Doing Something Useful with MySetup.wxs

Now let's do something useful and add an application to our solution. For this example, we will create a C# Windows Forms Application, but you can use whatever programming language you prefer.

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".
4. Be sure to choose the **Add to Solution** option in the **Solution** drop-down as depicted in Figure 2.

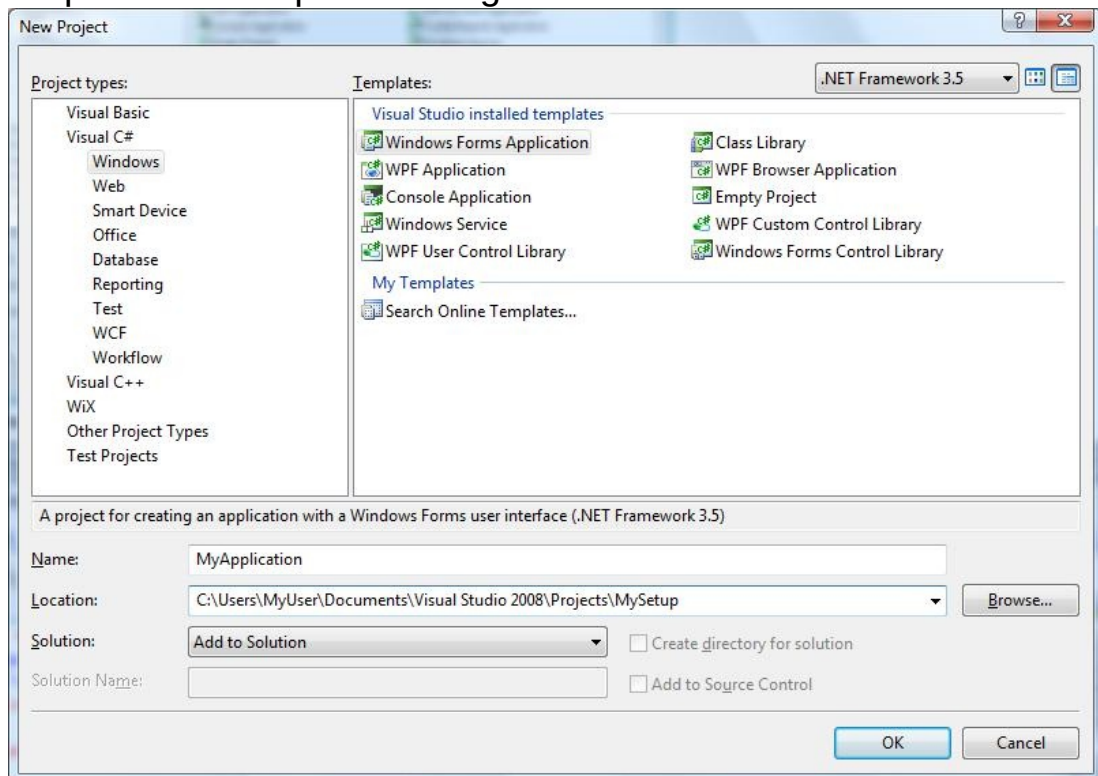


Figure 2. Creating MyApplication project in the solution

We need to make sure that the MyApplication project is built before the MySetup project because we will use the output from the MyApplication project build as an input into the setup build. For the purposes of this example, it does not matter what the application does, so we will not change the generated application code. No matter what the application

does, we need to get it to install via setup, so let's return to the MySetup project and the MySetup.wxs file to add this new application. While we are there, let's add a shortcut to the application to the installer. To create the appropriate project dependencies, right-click on the **References** node under the **MySetup** project and choose **Add Reference....** In that dialog, choose the **Projects** tab, click on the **MyApplication** project, and click the **Add** button as depicted in Figure 3.

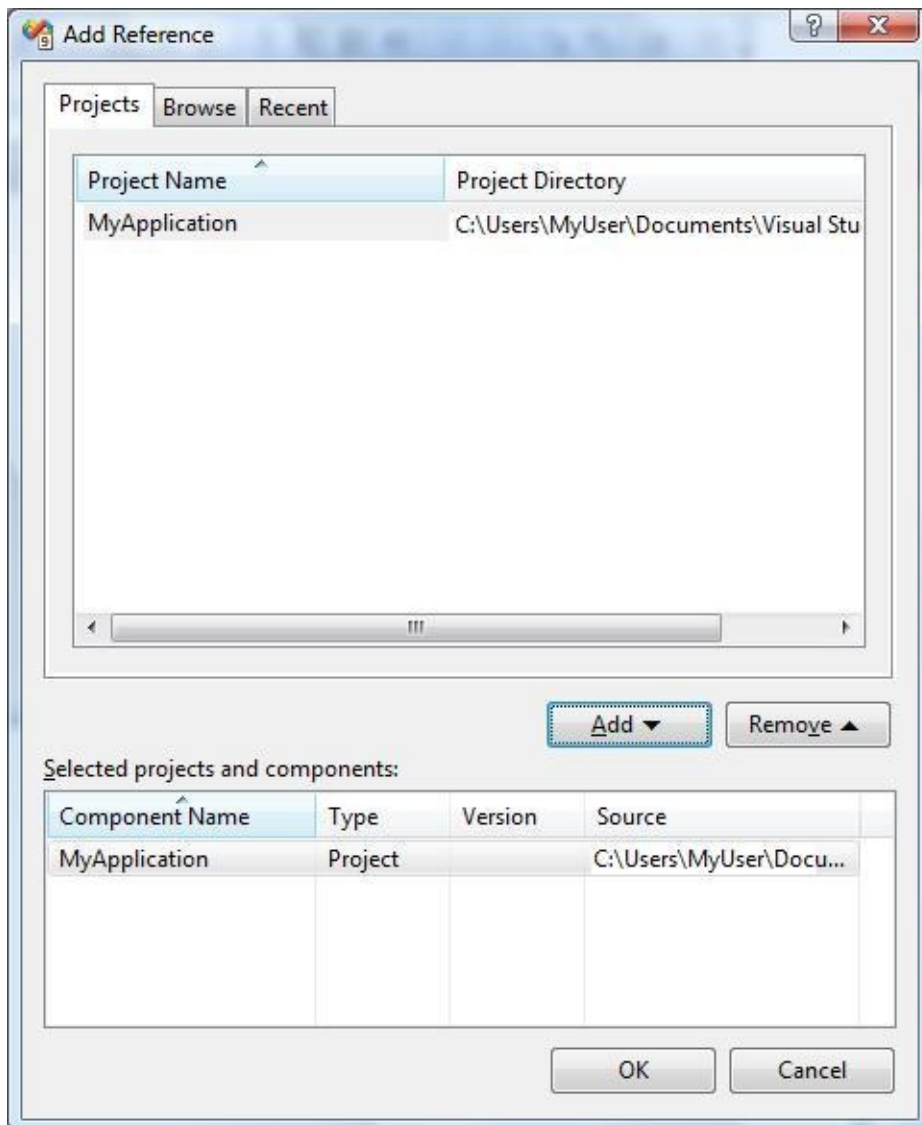


Figure 3. MySetup project references the output of the MyApplication project

Now we need to get this application to install via setup, so let's return to the MySetup project and the MySetup.wxs file to add this new application.

Open MySetup.wxs and you will see a comment that says:

```
<!-- TODO: Insert your files, registry keys, and other resource
```

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

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

Note: 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 [Adding Project References topic](#).

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.

Before building the solution again, we need to make sure that the MyApplication project is built before the MySetup project because the output from the MyApplication project build is an input into the setup build. To create the appropriate project dependencies, right-click on the **References** node under the **MySetup** project and choose **Add Reference...** In that dialog, choose the **Projects** tab, click on the

MyApplication project, click the **Add** button, and click **Ok**.

Rebuilding with these changes will create a setup package that can install and uninstall your application.

Adding Project References

Introduction

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 definitions which are set on the Candle command line and can be referenced in source files.

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, then click OK to dismiss the dialog.

List of Supported Project References

The WiX project supports the following project reference variables:

Variable name	Example usage	Element
<code>var.ProjectName.Configuration</code>	<code>\$(var.MyProject.Configuration)</code>	Element
<code>var.ProjectName.FullConfiguration</code>	<code>\$(var.MyProject.FullConfiguration)</code>	Element
<code>var.ProjectName.Platform</code>	<code>\$(var.MyProject.Platform)</code>	Attribute
<code>var.ProjectName.ProjectDir</code>	<code>\$(var.MyProject.ProjectDir)</code>	Child 2
<code>var.ProjectName.ProjectExt</code>	<code>\$(var.MyProject.ProjectExt)</code>	Child
<code>var.ProjectName.ProjectFileName</code>	<code>\$(var.MyProject.ProjectFileName)</code>	Member
<code>var.ProjectName.ProjectName</code>	<code>\$(var.MyProject.ProjectName)</code>	Member
<code>var.ProjectName.ProjectPath</code>	<code>\$(var.MyProject.ProjectPath)</code>	Child 2
<code>var.ProjectName.TargetDir</code>	<code>\$(var.MyProject.TargetDir)</code>	Child 2
<code>var.ProjectName.TargetExt</code>	<code>\$(var.MyProject.TargetExt)</code>	Child
<code>var.ProjectName.TargetFileName</code>	<code>\$(var.MyProject.TargetFileName)</code>	Member
<code>var.ProjectName.TargetName</code>	<code>\$(var.MyProject.TargetName)</code>	Member

<code>var.ProjectName.TargetPath</code>	<code>\$(var.MyProject.TargetPath)</code>	C 2
<code>var.ProjectName.Culture.TargetPath</code>	<code>\$(var.MyProject.en-US.TargetPath)</code>	C 2 L
<code>var.SolutionDir</code>	<code>\$(var.SolutionDir)</code>	C 2
<code>var.SolutionExt</code>	<code>\$(var.SolutionExt)</code>	.s
<code>var.SolutionFileName</code>	<code>\$(var.SolutionFileName)</code>	M
<code>var.SolutionName</code>	<code>\$(var.SolutionName)</code>	M
<code>var.SolutionPath</code>	<code>\$(var.SolutionPath)</code>	C 2

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

Example

The following File element demonstrates how to use project references in WiX authoring:

```
<File Id="MyExecutable" Name="$(var.MyProject.TargetFileName)" Sour
```

Adding WiX References

Introduction

The WiX toolset supports adding WiX library and extension DLL references to a WiX project. This allows the reuse of WiX elements (such as Custom Actions and Properties) defined in those references. When a WiX reference is added to the project, the WiX toolset automatically adds the necessary parameters to the compiler and linker command lines so that it will be correctly resolved when building the project.

To add a WiX library or extension 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 Browse tab.
3. Locate the desired .wixlib files and/or WiX extensions and click the Add button, then click OK to dismiss the dialog.

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](#)

Creating a .wixproj File

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:

```
<Project DefaultTargets="Build" xmlns="http://schemas.microsoft.com
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == ''
    <Platform Condition=" '$(Platform)' == '' ">x86</Pl
    <ProductVersion>3.0</ProductVersion>
    <ProjectGuid>{c523055d-a9d0-4318-ae85-ec934d33204b}</ProjectGuid>
    <SchemaVersion>2.0</SchemaVersion>
    <OutputName>WixProject1</OutputName>
    <OutputType>Package</OutputType>
    <WixTargetsPath Condition=" '$(WixTargetsPath)' ==
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' =
    <OutputPath>bin\$(Configuration)\</OutputPath>
    <IntermediateOutputPath>obj\$(Configuration)\</Inte
    <DefineConstants>Debug</DefineConstants>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' =
    <OutputPath>bin\$(Configuration)\</OutputPath>
    <IntermediateOutputPath>obj\$(Configuration)\</Inte
  </PropertyGroup>
  <ItemGroup>
    <Compile Include="Product.wxs" />
  </ItemGroup>
  <Import Project="$(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.
4. Copy the contents of **c:\Program Files\MSBuild\Microsoft\WiX\v3.0** 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\sdksdk** 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:

```
<PropertyGroup>
  <WixToolPath>$(SourceCodeControlRoot)\wix\3.0.4311.0\</WixT
  <WixTargetsPath>$(WixToolPath)wix.targets</WixTargetsPath>
  <WixTasksPath>$(WixToolPath)wixtasks.dll</WixTasksPath>
</PropertyGroup>
```

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 **..\tools**), 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 [WiX project file](#), 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.
3. Add the following build configurations to the <ConfigurationToBuild> section if they do not already exist there, or update them if they do already exist:

```
<ConfigurationToBuild Include="Debug|Mixed Platforms">
  <FlavorToBuild>Debug</FlavorToBuild>
  <PlatformToBuild>Mixed Platforms</PlatformToBuild>
</ConfigurationToBuild>
<ConfigurationToBuild Include="Release|Mixed Platforms">
  <FlavorToBuild>Release</FlavorToBuild>
  <PlatformToBuild>Mixed Platforms</PlatformToBuild>
</ConfigurationToBuild>
```

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](#)

[Light Task](#)

[Lit Task](#)

Candle Task

The Candle task wraps [candle.exe](#), 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:

```
<PropertyGroup>
  <CompilerTreatWarningsAsErrors>False</CompilerTreatWarningsAsEr
  <CompilerVerboseOutput>True</CompilerVerboseOutput>
  <DefineConstants>Variable1=value1;Variable2=value2</DefineConst
  <InstallerPlatform>x86</InstallerPlatform>
  <SuppressSpecificWarnings>1111</SuppressSpecificWarnings>
  <TreatSpecificWarningsAsErrors>2222</TreatSpecificWarningsAsErr
</PropertyGroup>
```

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.

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

	d<name>[=<value>] switch in candle.exe.
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.
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.
InstallerPlatform	Optional string parameter.

	<p>Specifies the processor architecture for the package. Valid values are x86, intel, x64, intel64 or ia64. This is equivalent to the -arch switch in candle.exe.</p>
OnlyValidateDocuments	<p>Optional boolean parameter.</p> <p>Specifies that the compiler should only validate documents. This is equivalent to the -zs switch in candle.exe.</p>
Pedantic	<p>Optional boolean parameter.</p> <p>Specifies that the compiler should display pedantic messages. This is equivalent to the -pedantic switch in candle.exe.</p>
ShowSourceTrace	<p>Optional boolean parameter.</p> <p>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.</p>

Light Task

The Light task wraps [light.exe](#), 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:

```
<PropertyGroup>  
  <LinkerTreatWarningsAsErrors>False</LinkerTreatWarningsAsErrors>  
  <LinkerVerboseOutput>True</LinkerVerboseOutput>  
  <SuppressIces>ICE18;ICE45;ICE82</SuppressIces>  
  <SuppressSpecificWarnings>1111</SuppressSpecificWarnings>  
  <TreatSpecificWarningsAsErrors>2222</TreatSpecificWarningsAsErr<br>  
  <WixVariables>Variable1=value1;Variable2=value2</WixVariables>  
</PropertyGroup>
```

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.
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	<p>Optional boolean parameter.</p> <p>Specifies that the tool should display pedantic messages. This is equivalent to the -pedantic switch.</p>
SuppressAllWarnings	<p>Optional boolean parameter.</p> <p>Specifies that all warnings should be suppressed. This is equivalent to the -sw switch.</p>
SuppressIntermediateFileVersionMatching	<p>Optional boolean parameter.</p> <p>Specifies that the tool should suppress intermediate file version mismatch checking. This is equivalent to the -sv switch.</p>
SuppressSchemaValidation	<p>Optional boolean parameter.</p> <p>Specifies that schema validation of documents should be suppressed. This is equivalent to the -ss switch.</p>
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 **Light** task.

Parameter	Description
AllowIdenticalRows	Optional boolean parameter. Specifies that the lir

	<p>should allow identical rows. Identical rows be treated as warnings. This is equivalent to the <code>-ai</code> switch in <code>light.exe</code>.</p>
AllowUnresolvedReferences	<p>Optional boolean parameter.</p> <p>Specifies that the linker should allow unresolved references. This will create valid output. This is equivalent to the <code>-ai</code> switch in <code>light.exe</code>.</p>
AdditionalCub	<p>Optional string parameter.</p> <p>Specifies an additional <code>.cub</code> file that the linker should use when running ICE validation. This is equivalent to the <code>-cu <file.cub></code> switch in <code>light.exe</code>.</p>
BackwardsCompatibleGuidGeneration	<p>Optional boolean parameter.</p> <p>Specifies that the linker should use the backwards compatible GUID generation algorithm. This is equivalent to the <code>-bcg</code> switch in <code>light.exe</code>.</p>
CabinetCachePath	<p>Optional string parameter.</p> <p>Specifies a path that the linker should use to</p>

	<p>built cabinet files. This is equivalent to the -cc <path> switch in light.exe.</p>
CabinetCreationThreadCount	<p>Optional integer parameter.</p> <p>Specifies the number of threads that the linker should use when building cabinet files. This is equivalent to the -ct switch in light.exe.</p>
Cultures	<p>Optional string parameter.</p> <p>Specifies a semicolon-comma delimited list of localized string cultures to load from .wxl files and libraries. Precedence of cultures is from left to right. This is equivalent to the -cultures:<culture> switch in light.exe.</p>
DefaultCompressionLevel	<p>Optional string parameter.</p> <p>Specifies the compression level that the linker should use when building cabinet files. Valid values are low, medium, high, none, or mszip. This is equivalent to the -dcl:<level> switch in light.exe.</p>
DropUnrealTables	<p>Optional boolean parameter.</p>

	<p>Specifies that the linker should drop unreal time from the output image. This is equivalent to the <code>-d</code> switch in <code>light.exe</code>.</p>
Ices	<p>Optional string parameter.</p> <p>Specifies that the linker should run specific internal consistency evaluators (ICES). This is equivalent to the <code>-ic <ICE></code> switch in <code>light.exe</code>.</p>
LeaveTemporaryFiles	<p>Optional boolean parameter.</p> <p>Specifies that the linker should not delete temporary files. This is equivalent to the <code>-n</code> switch in <code>light.exe</code>.</p>
LinkerAdditionalOptions	<p>Optional string parameter.</p> <p>Specifies additional command line parameters to append when calling <code>light.exe</code>.</p>
LinkerBaseInputPaths	<p>Optional string parameter.</p> <p>Specifies a base path the linker should use to locate all files. This is equivalent to the <code>-b <path></code> switch in <code>light.exe</code>.</p>

LinkerBindFiles	<p>Optional boolean parameter.</p> <p>Specifies that the linker 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 in light.exe.</p>
LinkerPedantic	<p>Optional boolean parameter.</p> <p>Specifies that the linker should display pedantic messages. This is equivalent to the -pedantic switch in light.exe.</p>
LinkerSuppressAllWarnings	<p>Optional boolean parameter.</p> <p>Specifies that all linker warnings should be suppressed. This is equivalent to the -sv switch in light.exe.</p>
LinkerSuppressIntermediateFileVersionMatching	<p>Optional boolean parameter.</p> <p>Specifies that the linker should suppress intermediate file version mismatch checking. This is equivalent to the - switch in light.exe.</p>

LinkerSuppressSchemaValidation	<p>Optional boolean parameter.</p> <p>Specifies that the linker should suppress schema validation of documents. This is equivalent to the <code>/ss</code> switch in <code>light.exe</code>.</p>
LinkerSuppressSpecificWarnings	<p>Optional string parameter.</p> <p>Specifies that certain linker warnings should be suppressed. This is equivalent to the <code>-sw</code> switch in <code>light.exe</code>.</p>
LinkerTreatSpecificWarningsAsErrors	<p>Optional string parameter.</p> <p>Specifies that certain linker warnings should be treated as errors. This is equivalent to the <code>-w</code> switch in <code>light.exe</code>.</p>
LinkerTreatWarningsAsErrors	<p>Optional boolean parameter.</p> <p>Specifies that all linker warnings should be treated as errors. This is equivalent to the <code>-w</code> switch in <code>light.exe</code>.</p>
LinkerVerboseOutput	<p>Optional boolean parameter.</p> <p>Specifies that the linker should provide verbose output. This is equivalent to the <code>/v</code> switch in <code>light.exe</code>.</p>

	to the -v switch in light.exe.
OutputAsXml	<p>Optional boolean parameter.</p> <p>Specifies that the lir should output a .wix file instead of a .msi. This is equivalent to the -xo switch in light.exe.</p>
PdbOutputFile	<p>Optional string parameter.</p> <p>Specifies that the lir should create the output .wixpdb file with the provided name. This is equivalent to the -p <output.wixpdb> switch in light.exe.</p>
ReuseCabinetCache	<p>Optional boolean parameter.</p> <p>Specifies that the lir should reuse cabinet files from the cabinet cache. This is equivalent to the -reusecab switch in light.exe.</p>
SetMsiAssemblyNameFileVersion	<p>Optional boolean parameter.</p> <p>Specifies that the lir should add a fileVersion entry to the MsiAssemblyName for each assembly. This is equivalent to the -fv</p>

	switch in light.exe.
SuppressAclReset	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress res ACLs. This is useful laying out an image network share. This is equivalent to the -sa switch in light.exe.</p>
SuppressAssemblies	<p>Optional boolean parameter.</p> <p>Specifies that the lir should not get asse name information fo assemblies. This is equivalent to the -sa switch in light.exe.</p>
SuppressDefaultAdminSequenceActions	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress de admin sequence ac This is equivalent to sadmin switch in lig</p>
SuppressDefaultAdvSequenceActions	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress de advertised sequenc actions. This is equi to the -sadv switch i light.exe.</p>

<p>SuppressDefaultUISequenceActions</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress de UI sequence actions is equivalent to the switch in light.exe.</p>
<p>SuppressFileHashAndInfo</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress gathering file inform (hash, version, lang etc). This is equivalent the -sh switch in light</p>
<p>SuppressFiles</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress gathering all file data. This has the same effect as setting the SuppressAssemblies SuppressFileHashAndInfo parameters. This is equivalent to the -sf switch in light.exe.</p>
<p>SuppressIces</p>	<p>Optional string parameter.</p> <p>Specifies that the lir should suppress running specific ICES. This is equivalent to the -si <ICE> switch in light</p>

<p>SuppressLayout</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress lay creation. This is equivalent to the -sl switch in light.exe.</p>
<p>SuppressMsiAssemblyTableProcessing</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress processing the data MsiAssembly table. is equivalent to the switch in light.exe.</p>
<p>SuppressPdbOutput</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress outputting .wixpdb fi This is equivalent to spdb switch in light.</p>
<p>SuppressValidation</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress .m .msm validation. Th equivalent to the -sv switch in light.exe.</p>
<p>SuppressTagSectionIdAttributeOnTuples</p>	<p>Optional boolean parameter.</p> <p>Specifies that the lir should suppress ad</p>

	<p>the sectionId attribute. This is equivalent to the -sts switch in light.exe.</p>
UnreferencedSymbolsFile	<p>Optional string parameter.</p> <p>Specifies an unreferenced symbols file that the installer should use. This is equivalent to the -us <output.xml> switch in light.exe.</p>
WixVariables	<p>Optional string parameter.</p> <p>Specifies a semicolon-delimited list of bindable WiX variables. This is equivalent to the -d<name>[=<value>] switch in light.exe.</p>

Lit Task

The Lit task wraps [lit.exe](#), 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:

```
<PropertyGroup>  
  <LibTreatWarningsAsErrors>False</LibTreatWarningsAsErrors>  
  <LibVerboseOutput>True</LibVerboseOutput>  
  <SuppressSpecificWarnings>1111</SuppressSpecificWarnings>  
  <TreatSpecificWarningsAsErrors>2222</TreatSpecificWarningsAsErr<br></PropertyGroup>
```

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.

<p>SuppressAllWarnings</p>	<p>Optional boolean parameter.</p> <p>Specifies that all warnings should be suppressed. This is equivalent to the -sw switch.</p>
<p>SuppressIntermediateFileVersionMatching</p>	<p>Optional boolean parameter.</p> <p>Specifies that the tool should suppress intermediate file version mismatch checking. This is equivalent to the -sv switch.</p>
<p>SuppressSchemaValidation</p>	<p>Optional boolean parameter.</p> <p>Specifies that schema validation of documents should be suppressed. This is equivalent to the -ss switch.</p>
<p>SuppressSpecificWarnings</p>	<p>Optional string parameter.</p> <p>Specifies that certain warnings should be suppressed. This is equivalent to the -sw[N] switch.</p>
<p>TreatSpecificWarningsAsErrors</p>	<p>Optional string parameter.</p> <p>Specifies that certain warnings should be</p>

	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 **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 .wixout file. This is only valid when the

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

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

	<p>Specifies that the library creation tool should provide verbose output. This is equivalent to the -v switch in lit.exe.</p>
LinkerBaseInputPaths	<p>Optional string parameter.</p> <p>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.</p>

Using WiX With NAnt

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

```
<!-- WiX 3 folder -->  
<property name="wix.dir" value="{path::combine(environment::get-va  
<!-- Load the WiX3 tasks -->  
<loadtasks assembly="{wix.dir}\Microsoft.Tools.WindowsInstallerXml
```

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 [candle.exe](#), 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.
includedirs	Optional NAnt fileset . Specifies directories to add to the compiler include search path. This is equivalent to the -I<dir> switch in candle.exe.

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

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

Light Task

The Light task wraps [light.exe](#), 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:

```
<light
  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>
</light>
```

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 <i>"HKLM\SOFTWARE\Microsoft\Windows Installer XML\3.0\InstallRoot"</i> 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 <code>-out</code> 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 (.wxi) 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.
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 [lit.exe](#), 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:

```
<lit
  exedir="${wix.dir}"
  out="${output.dir}\Setup.wixlib"
  bindfiles="true"
  rebuild="true">
  <sources basedir="${release.dir}\Setup\obj">
    <include name="*.wixobj" />
  </sources>
</lit>
```

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 <i>"HKLM\SOFTWARE\Microsoft\Windows Installer XML\3.0\InstallRoot"</i> 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 **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 (.wxi) 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

[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: Get the parent directory of a file search](#)

Redistributables and Install Checks

[How To: Check for .NET Framework versions](#)

[How To: Install the .NET Framework using a bootstrapper](#)

[How To: Install DirectX 9.0 with your installer](#)

[How To: Install the Visual C++ Redistributable with your installer](#)

[How To: Block installation based on OS version](#)

User Interface and Localization

[How To: Build a localized version of your installer](#)

[How To: Make your installer localizable](#)

[How To: Run the installed application after setup](#)

[How To: Set your installer's icon in Add/Remove Programs](#)

Updates

[How To: Implement a major upgrade in your installer](#)

General How Tos

[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: 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.

```
<Directory Id="TARGETDIR" Name="SourceDir">
  <Directory Id="ProgramFilesFolder">
    <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Applicati
  </Directory>
</Directory>
```

The element with the id [TARGETDIR](#) 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 [ProgramFilesFolder](#), 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 [public property](#) 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.

```
<DirectoryRef Id="APPLICATIONROOTDIRECTORY">
  <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
    <File Id="myapplication.exe" Source="MySourceFiles\MyApplic
  </Component>
  <Component Id="documentation.html" Guid="PUT-GUID-HERE">
    <File Id="documentation.html" Source="MySourceFiles\documen
  </Component>
</DirectoryRef>
```

The [<DirectoryRef>](#) 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 [<ComponentRef>](#) 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" L
    <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 A
        </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\
        </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE"
        <File Id="documentation.html" Source="MySourceFiles
        </Component>
    </DirectoryRef>

    <!-- Step 3: Tell WiX to install the files -->
    <Feature Id="MainApplication" Title="Main Application" Leve
      <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.

```
<Property Id="USER32VERSION">
  <DirectorySearch Id="SystemFolderDriverVersion" Path="[SystemFo
    <FileSearch Name="user32.dll" MinVersion="6.0.6001.1750"/>
  </DirectorySearch>
</Property>
```

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
  <![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.

```
<DirectoryRef Id="TARGETDIR">
  <Component Id="RegistryEntries" Guid="PUT-GUID-HERE">
    <RegistryKey Root="HKCU"
      Key="Software\Microsoft\MyApplicationName"
      Action="createAndRemoveOnUninstall">
      <RegistryValue Type="integer" Name="SomeIntegerValue" V
      <RegistryValue Type="string" Value="Default Value"/>
    </RegistryKey>
  </Component>
</DirectoryRef>
```

The snippet begins with a `DirectoryRef` that points to the `TARGETDIR` 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 `<RegistryKey>` element to specify where in the registry the values should go. In this example the key is under

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*.

```
<Property Id="NETFRAMEWORK20">
  <RegistrySearch Id="NetFramework20"
    Root="HKLM"
    Key="Software\Microsoft\NET Framework Setup\NDP
    Name="Install"
    Type="raw" />
</Property>
```

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. P  
  <![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  
</Directory>
```

The [ProgramMenuFolder](#) 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.

```
<DirectoryRef Id="ApplicationProgramsFolder">
  <Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE">
    <Shortcut Id="ApplicationStartMenuShortcut"
      Name="My Application Name"
      Description="My Application Description"
      Target="[APPLICATIONROOTDIRECTORY]MyApplication.e
      WorkingDirectory="APPLICATIONROOTDIRECTORY"/>
    <RemoveFolder Id="ApplicationProgramsFolder" On="uninstall"
    <RegistryValue Root="HKCU" Key="Software\Microsoft\MyApplic
  </Component>
</DirectoryRef>
```

The [<DirectoryRef>](#) 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>](#) element, then reference it 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" L
    <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 A
      </Directory>
      <!-- Step 1: Define the directory structure -->
      <Directory Id="ProgramMenuFolder">
        <Directory Id="ApplicationProgramsFolder" Name="My
      </Directory>
    </Directory>

    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE"
        <File Id="documentation.html" Source="MySourceFiles
      </Component>
    </DirectoryRef>

    <!-- Step 2: Add the shortcut to your installer package -->
    <DirectoryRef Id="ApplicationProgramsFolder">
      <Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE"
        <Shortcut Id="ApplicationStartMenuShortcut"
          Description="My Application Description"
          Target="[APPLICATIONROOTDIRECTORY]MyApplication
            WorkingDirectory="APPLICATIONROOTDIRECTOR
          <RemoveFolder Id="ApplicationProgramsFolder" On="un
          <RegistryValue Root="HKCU" Key="Software\Microsoft\
        </Component>
    </DirectoryRef>

    <Feature Id="MainApplication" Title="Main Application" Leve
      <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 [<Wix>](#) element in your project by adding the following attribute:

```
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 [<Util:InternetShortcut>](#) element. The following example adds an InternetShortcut element to the existing shortcut creation example from [How To: Create a shortcut on the Start Menu](#).

```
<DirectoryRef Id="ApplicationProgramsFolder">
  <Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE">
    <Shortcut Id="ApplicationStartMenuShortcut"
      Name="My Application Name"
      Description="My Application Description"
      Target="[APPLICATIONROOTDIRECTORY]MyApplication.e
      WorkingDirectory="APPLICATIONROOTDIRECTORY"/>
    <util:InternetShortcut Id="OnlineDocumentationShortcut"
      Name="My Online Documentation"
      Target="http://www.wixwiki.com/" />
    <RemoveFolder Id="ApplicationProgramsFolder" On="uninstall"
    <RegistryValue Root="HKCU" Key="Software\Microsoft\MyApplic
  </Component>
</DirectoryRef>
```

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 [How To: Create a Shortcut on the Start Menu](#) 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 [System64Folder](#) 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" L
    <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 A
      </Directory>
      <Directory Id="ProgramMenuFolder">
        <Directory Id="ApplicationProgramsFolder" Name="My
      </Directory>
    </Directory>

    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE"
        <File Id="documentation.html" Source="MySourceFiles
      </Component>
    </DirectoryRef>

    <DirectoryRef Id="ApplicationProgramsFolder">
      <Component Id="ApplicationShortcut" Guid="PUT-GUID-HERE
        <Shortcut Id="ApplicationStartMenuShortcut"
          Description="My Application Description"
          Target="[APPLICATIONROOTDIRECTORY]MyApplication
            WorkingDirectory="APPLICATIONROOTDIRECTOR
        <!-- Step 1: Add the uninstall shortcut to your ins
        <Shortcut Id="UninstallProduct"
          Name="Uninstall My Application"
          Description="Uninstalls My Application"
          Target="[System64Folder]msiexec.exe"
          Arguments="/x [ProductCode]"/>
        <RemoveFolder Id="ApplicationProgramsFolder" On="un
        <RegistryValue Root="HKCU" Key="Software\Microsoft\
      </Component>
```

```
</DirectoryRef>

  <Feature Id="MainApplication" Title="Main Application" Level="Default" >
    <ComponentRef Id="myapplication.exe" />
    <ComponentRef Id="documentation.html" />
    <ComponentRef Id="ApplicationShortcut" />
  </Feature>
</Product>
</Wix>
```

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:

```
<Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
  <File Id="myapplication.exe" Source="MySourceFiles\MyApplicatio
    <netfx:NativeImage Id="ngen_MyApplication.exe" Platform="32
  </File>
</Component>
```

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.

```
<Property Id="SHDOCVW">
  <DirectorySearch Id="winDir" Path="[WindowsFolder]">
    <DirectorySearch Id="Media" Path="Media">
      <FileSearch Id="Chimes" Name="chimes.wav" />
    </DirectorySearch>
  </DirectorySearch>
</Property>
```

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.

```
<Property Id="USER32">
  <DirectorySearchRef Id="Media" Parent="WinDir" Path="Media">
    <FileSearch Id="Chord" Name="chord.wav" />
  </DirectorySearchRef>
</Property>
```

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.

```
<Property Id="NGEN2DIR">  
  <DirectorySearch Id="Windows" Path="[WindowsFolder]">  
    <DirectorySearch Id="MS.NET" Path="Microsoft.NET">  
    </DirectorySearch>  
  </DirectorySearch>  
</Property>
```

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.

```
<Property Id="NGEN2DIR">
  <DirectorySearch Id="Windows" Path="[WindowsFolder]">
    <DirectorySearch Id="MS.NET" Path="Microsoft.NET">
      <DirectorySearch Id="Ngen2Dir" Depth="2" AssignToProperty="yes">
        <FileSearch Id="Ngen_exe" Name="ngen.exe" MinVersion="2.0.0.0">
        </FileSearch>
      </DirectorySearch>
    </DirectorySearch>
  </DirectorySearch>
</Property>
```

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 [<PropertyRef>](#) element. For each property you want to use, add the corresponding PropertyRef to your project. For example, if you are interested in detecting .NET Framework 2.0 add the following:

```
<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. P
  <![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 .NET Framework 2.0 is 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 SP
  <![CDATA[Installed OR (NETFRAMEWORK30_SP_LEVEL and NOT NETFRAME
</Condition>
```

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 then 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 [Visual C++](#) and [DirectX](#) 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**
3. In **Solution Explorer** right click on your project file and select **Edit <projectname>**

Step 2: Add items for prerequisites

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

```
<ItemGroup>
  <BootstrapperFile Include="Microsoft.Net.Framework.3.5">
    <ProductName>.NET Framework 3.5</ProductName>
  </BootstrapperFile>
  <BootstrapperFile Include="Microsoft.Windows.Installer.3.1">
    <ProductName>Windows Installer 3.1</ProductName>
  </BootstrapperFile>
</ItemGroup>
```

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:

```
<Target Name="AfterBuild">
  <GenerateBootstrapper ApplicationFile="$(TargetFileName)"
    ApplicationName="My Application Name"
    BootstrapperItems="@(\BootstrapperFile)"
    ComponentsLocation="Relative"
    CopyComponents="True"
    OutputPath="$(OutputPath)"
    Path="C:\Program Files\Microsoft SDKs\Win
</Target>
```

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 pre-requisite 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 [Bootstrapper Manifest Generator](#) 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 [Installing DirectX with DirectSetup](#) 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 [How To: Add a file to your installer](#). 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="yes"/>
      <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.dl
      <File Id="DEC2006_d3dx9_32_x86.cab"
        Source="MySourceFiles\DirectXMinInstall\DEC2006_d3dx9_32
    </Component>
  </Directory>
</DirectoryRef>

<Feature Id="DirectXRedist"
  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 **/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 `<UI>` and `<ProgressText>` elements are used, as illustrated in the following example.

```
<UI>  
  <ProgressText Action="InstallDirectX">Installing DirectX 9.0c</  
</UI>
```

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 [<Merge>](#) and [<MergeRef>](#) elements. The following example illustrates how these elements are used.

```
<DirectoryRef Id="TARGETDIR">  
  <Merge Id="VCRedist" SourceFile="MySourceFiles\Microsoft_VC80_C  
</DirectoryRef>
```

```
<Feature Id="VCRedist" Title="Visual C++ 8.0 Runtime" AllowAdvertis  
  <MergeRef Id="VCRedist"/>  
</Feature>
```

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 DiskId attribute should match the DiskId 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 t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE03: String overflow (greater t
light.exe(0,0): warning LGHT1076: ICE25: Possible dependency failur
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.9
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.9
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.9
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.9
light.exe(0,0): warning LGHT1076: ICE82: This action SystemFolder.9
```

These warnings are expected and are due to how the Visual C++ merge modules were authored. For more details see [Aaron Stebner's blog entry](#).

How To: Block Installation Based on OS Version

Windows Installer provides the standard [VersionNT](#) 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 V
    <![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 [VersionNT64](#) property. To check for versions of Windows prior to Windows NT use the [Windows9X](#) property.

How To: User Interface and Localization

This section includes guides for building installer UI and localizing your installer.

[How To: Build a localized version of your installer](#)

[How To: Make your installer localizable](#)

[How To: Run the installed application after setup](#)

[How To: Set your installer's icon in Add/Remove Programs](#)

How To: Build a Localized Version of Your Installer

Once you have described all the strings in your installer using language files, as described in [How To: Make your installer localizable](#), you can then build versions of your installer for each supported language. This how to explains building the localized installers both from the command line and using Visual Studio.

Option 1: Building localized installers from the command line

The first step in building a localized installer is to compile your WiX sources using candle.exe:

```
candle.exe myinstaller.wxs -out myinstaller.wixobj
```

After the intermediate output file is generated you can then use light.exe to generate multiple localized MSIs:

```
light.exe myinstaller.wixobj -cultures:en-us -loc en-us.wxl -out my  
light.exe myinstaller.wixobj -cultures:fr-fr -loc fr-fr.wxl -out my
```

The -loc flag is used to specify the language file to use. It is important to include the -cultures flag on the command line to ensure the correct localized strings are included for extensions such as [WiXUIExtension](#).

Option 2: Building localized installers using Visual Studio

Visual Studio will automatically build localized versions of your installer. If your WiX project includes multiple .wxi files, one localized installer will be built for each culture, unless **Cultures to build** is specified.

For more information, see [Specifying cultures to build](#)

How To: Make your installer localizable

WiX supports building localized installers through the use of language files that include localized strings. It is a good practice to put all your strings in a language file as you create your setup, even if you do not currently plan on shipping localized versions of your installer. This how to describes how to create a language file and use its strings in your WiX project.

Step 1: Create the language file

Language files end in the .wxl extension and specify their culture using the [<WixLocalization>](#) element. To create a language file on the command line create a new file with the appropriate name and add the following:

```
<?xml version="1.0" encoding="utf-8"?>
<WixLocalization Culture="en-us" xmlns="http://schemas.microsoft.co
</WixLocalization>
```

If you are using Visual Studio you can add a new language file to your project by doing the following:

1. Right click on your project in Solution Explorer and select Add > New Item...
2. Select WiX Localization File, give the file an appropriate name, and select Add

By default Visual Studio creates language files in the en-us culture. To create a language file for a different culture change the Culture attribute to the appropriate culture string.

Step 2: Add the localized strings

Localized strings are defined using the [<String>](#) element. Each element consists of a unique id for later reference in your WiX project and the string value. For example:

```
<String Id="ApplicationName">My Application Name</String>  
<String Id="ManufacturerName">My Manufacturer Name</String>
```

The String element goes inside the WixLocalization element, and you should add one String element for each piece of text you need to localize.

Step 3: Use the localized strings in your project

Once you have defined the strings you can use them in your project wherever you would normally use text. For example, to set your product's Name and Manufacturer to the localized strings do the following:

```
<Product Id="*"
    UpgradeCode="PUT-GUID-HERE"
    Version="1.0.0.0"
    Language="1033"
    Name="!(loc.ApplicationName)"
    Manufacturer="!(loc.ManufacturerName)">
```

Localization strings are referenced using the **!(loc.stringname)** syntax. These references will be replaced with the actual strings for the appropriate locale at build time.

For information on how to compile localized versions of your installer once you have the necessary language files see [How To: Build a localized version of your installer](#).

How To: Run the Installed Application After Setup

Often when completing the installation of an application it is desirable to offer the user the option of immediately launching the installed program when setup is complete. This how to describes customizing the default WiX UI experience to include a checkbox and a WiX custom action to launch the application if the checkbox is checked.

This how to assumes you have already created a basic WiX project using the steps outlined in [How To: Add a file to your installer](#).

Step 1: Add the extension libraries to your project

This walkthrough requires WiX extensions for UI components and custom actions. These extension libraries 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 WixUIExtension -ext WixUtilExtension
```

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

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

Step 2: Add UI to your installer

The WiX [Minimal UI](#) sequence includes a basic set of dialogs that includes a finished dialog with optional checkbox. To include the sequence in your project add the following snippet anywhere inside the <Product> element.

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

To display the checkbox on the last screen of the installer include the following snippet anywhere inside the <Product> element:

```
<Property Id="WIXUI_EXITDIALOGOPTIONALCHECKBOXTEXT" Value="Launch M
```

The WIXUI_EXITDIALOGOPTIONALCHECKBOXTEXT property is provided by the standard UI sequence that, when set, displays the checkbox and uses the specified value as the checkbox label.

Step 3: Include the custom action

Custom actions are included in a WiX project using the [<CustomAction>](#) element. Running an application is accomplished with the WixShellExecTarget custom action. To tell Windows Installer about the custom action, and to set its properties, include the following in your project anywhere inside the <Product> element:

```
<Property Id="WixShellExecTarget" Value="[#myapplication.exe]" />  
<CustomAction Id="LaunchApplication" BinaryKey="WixCA" DllEntry="wi
```

The Property element sets the WixShellExecTarget to the location of the installed application. WixShellExecTarget is the property Id the WixShellExec custom action expects will be set to the location of the file to run. The Value property uses the special # character to tell WiX to look up the full installed path of the file with the id myapplication.exe.

The CustomAction element includes the action in the installer. It is given a unique Id, and the BinaryKey and DllEntry properties indicate the assembly and entry point for the custom action. The Impersonate property tells Windows Installer to run the custom action as the installing user.

Step 4: Trigger the custom action

Simply including the custom action, as in Step 3, isn't sufficient to cause it to run. Windows Installer must also be told when the custom action should be triggered. This is done by using the [<Publish>](#) element to add it to the actions run when the user clicks the Finished button on the final page of the UI dialogs. The Publish element should be included inside the <UI> element from Step 2, and looks like this:

```
<Publish Dialog="ExitDialog"  
  Control="Finish"  
  Event="DoAction"  
  Value="LaunchApplication">WIXUI_EXITDIALOGOPTIONALCHECKBOX = 1
```

The Dialog property specifies the dialog the Custom Action will be attached to, in this case the ExitDialog. The Control property specifies that the Finish button on the dialog triggers the custom action. The Event property indicates that a custom action should be run when the button is clicked, and the Value property specifies the custom action that was included in Step 3. The condition on the element prevents the action from running unless the checkbox from Step 2 was checked and the application was actually installed (as opposed to being removed or repaired).

The Complete Sample

```
<?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 Application Name"
    Manufacturer="My Manufacturer Name">
    <Package InstallerVersion="300" Compressed="yes"/>
    <Media Id="1" Cabinet="myapplication.cab" EmbedCab="yes" />

    <!-- The following three sections are from the How To: Add a Fi
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder">
        <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Appli
      </Directory>
    </Directory>

    <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
      <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
        <File Id="myapplication.exe" Source="MySourceFiles\MyAp
      </Component>
      <Component Id="documentation.html" Guid="PUT-GUID-HERE">
        <File Id="documentation.html" Source="MySourceFiles\doc
      </Component>
    </DirectoryRef>

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

    <!-- Step 2: Add UI to your installer / Step 4: Trigger the cus
    <UI>
      <UIRef Id="WixUI_Minimal" />
      <Publish Dialog="ExitDialog"
        Control="Finish"
        Event="DoAction"
        Value="LaunchApplication">WIXUI_EXITDIALOGOPTIONALCHECK
    </UI>
    <Property Id="WIXUI_EXITDIALOGOPTIONALCHECKBOXTEXT" Value="Laun

    <!-- Step 3: Include the custom action -->
```

```
<Property Id="WixShellExecTarget" Value="#myapplication.exe"
<CustomAction Id="LaunchApplication"
  BinaryKey="WixCA"
  DllEntry="WixShellExec"
  Impersonate="yes" />
</Product>
</Wix>
```

How To: Set Your Installer's Icon in Add/Remove Programs

Windows Installer supports a standard property, [ARPPRODUCTICON](#), that controls the icon displayed in Add/Remove Programs for your application. To set this property you first need to include the icon in your installer using the [<Icon>](#) element, then set the property using the [<Property>](#) element.

```
<Icon Id="icon.ico" SourceFile="MySourceFiles\icon.ico"/>  
<Property Id="ARPPRODUCTICON" Value="icon.ico" />
```

These two elements can be placed anywhere in your WiX project under the Project element. There is no need to nest them in a Directory element. The Icon tag specifies the location of the icon on your source machine, and gives it a unique id for use later in the WiX project. The Property element sets the ARPPRODUCTION property to the id of the icon to use.

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 [Windows Installer major upgrades](#). 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 [UpgradeVersion](#) 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:

```
<Upgrade Id="PUT-GUID-HERE">
  <UpgradeVersion Minimum="1.0.0"
    IncludeMinimum="yes"
    Maximum="1.0.1"
    Property="OLDERVERSIONBEINGUPGRADED" />
</Upgrade>
```

The exact name of the property specified in the [UpgradeVersion](#) 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 [RemoveExistingProducts](#) action, you will see an error like the following:

```
error LGHT0094 : Unresolved reference to symbol 'WixAction:Installe
```

Windows Installer looks for other installed .msi files with the same UpgradeCode value during the [FindRelatedProducts](#) action. If you do not specifically schedule the [FindRelatedProducts](#) 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 [UpgradeVersion](#) 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:

```
<Upgrade Id="PUT-GUID-HERE">  
  <UpgradeVersion Minimum="1.0.1"  
    OnlyDetect="yes"  
    Property="NEWERVERSIONDETECTED" />  
</Upgrade>
```

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 ins  
  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 [WixUtilExtension](#) to use the WixExitEarlyWithSuccess custom action, either by adding it to the references list for your project if you are using [Votive and Visual Studio](#), or by passing it into [light.exe](#) 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 [Product](#) 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 [enable Windows Installer logging via the registry](#).

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 [Windows Installer 4.5 SDK](#), 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 [guidgen tool](#) that ships with Visual Studio, generally located under **Tools > Create GUID** menu, or the [GuidGen.com](#) 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 [change the letters in the GUID to uppercase](#) 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 newly-generated GUID.

The [<Component>](#), [<Package>](#), [<Patch>](#), [<Product>](#) 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.

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](#)

[Iis schema for WixIisExtension](#)

[IsolatedApp schema for WixIsolatedAppExtension](#)

[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"?>`
`<Include 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)

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

LaunchCondition table and evaluates each conditional statement recorded there.

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

AdminUISequence Element

Description

None

Windows Installer references

[AdminUISequence Table](#)

Parents

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

Inner Text

None

Children

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

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

volumes have enough space for the installation.

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

AdvertiseExecuteSequence Element

Description

None

Windows Installer references

[AdvExecuteSequence Table](#)

Parents

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

Inner Text

None

Children

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

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

table.

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

AllocateRegistrySpace Element

Description

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

Windows Installer references

[AllocateRegistrySpace Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

AppData Element

Description

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

Windows Installer references

None

Parents

[Category](#)

See Also

[Wix Schema](#)

Version 3.0.5419.0

AppId Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	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 AppId table row.	

Set this value to 'no' in order to generate Registry rows that perform similar registration (without the often problematic Windows Installer advertising behavior).

Description	String	Set this value to the description of the Appld. It can only be specified when the Appld is not being advertised.
DllSurrogate	String	Set this value to specify that the class is a DLL that is to be activated in a surrogate EXE process, and the surrogate process to be used is the path of a surrogate EXE file specified by the value.
LocalService	String	Set this value to the name of a service to allow the object to be installed as a Win32 service.
RemoteServerName	String	Set this value to the name of the remote server to configure the client

to request the object be run at a particular machine whenever an activation function is called for which a COSERVERINFO structure is not specified.

RunAsInteractiveUser	YesNoType	Set this value to 'yes' to configure a class to run under the identity of the user currently logged on and connected to the interactive desktop when activated by a remote client without being written as a Win32 service.
ServiceParameters	String	Set this value to the parameters to be passed to a LocalService on invocation.

Remarks

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

Id

In General

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

Specific Example

[HKCR\AppID\{**01234567-89AB-CDEF-0123-**

456789ABCDEF}}

ActivateAtStorage

In General

[HKCR\AppID\{Id}]

ActivateAtStorage="**ActivateAtStorage**"

Specific Example

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

ActivateAtStorage="Y"

Description

In General

[HKCR\AppID\{Id}]

@="**Description**"

Specific Example

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

@="My Appld Description"

DllSurrogate

In General

[HKCR\AppID\{Id}]

DllSurrogate="**DllSurrogate**"

Specific Example

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

DllSurrogate="C:\surrogate.exe"

LocalService

In General

[HKCR\AppID\{Id}]

LocalService="**LocalService**"

Specific Example

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

LocalService="MyServiceName"

RemoteServerName

In General

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

Specific Example

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

RunAsInteractiveUser

In General

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

Specific Example

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

ServiceParameters

In General

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

Specific Example

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

See Also

[Wix Schema](#)

AppSearch Element

Description

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

Windows Installer references

[AppSearch Table](#), [AppSearch Action](#)

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#), [ComponentSearch](#), [FileSearch](#), [IniFileSearch](#), [RegistrySearch](#)

Version 3.0.5419.0

AssemblyName Element

Description

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

Windows Installer references

[MsiAssemblyName Table](#)

Parents

[File](#)

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

Billboard Element

Description

Billboard to display during install of a Feature

Windows Installer references

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

Parents

[BillboardAction](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

BillboardAction Element

Description

Billboard action during which child Billboards are displayed

Windows Installer references

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

Parents

[UI](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

Binary Element

Description

Binary data used for CustomAction elements and UI controls.

Windows Installer references

[Binary Table](#)

Parents

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

Inner Text

None

Children

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

- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

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

		SourceFile attribute instead.
SuppressModularization	YesNoType	Use to suppress modularization of this Binary identifier in merge modules.
<hr/>		
Any attribute namespace='##other' processContents='lax'		

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

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

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

See Also

[Wix Schema](#)

Category Element

Description

Qualified published component for parent Component

Windows Installer references

[PublishComponent Table](#)

Parents

[Component](#)

Inner Text

None

Children

Sequence (min: 0, max: unbounded)

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

Attributes

Name	Type	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 in the Id attribute. A qualifier is used to distinguish multiple forms of the same Component, such as	Yes

a Component that is implemented
in multiple languages.

See Also

[Wix Schema](#)

Version 3.0.5419.0

CCPSearch Element

Description

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

Windows Installer references

[CCPSearch Action](#)

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#), [RMCCPSearch](#), [ComplianceCheck](#)

Version 3.0.5419.0

Class Element

Description

COM Class registration for parent Component.

Windows Installer references

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

Parents

[AppId](#), [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).
- [ProgId](#) (min: 0, max: unbounded): A ProgId associated with Class must be a child element of the Class element

Attributes

Name	Type	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).
AppId	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 AppId/@Id of an AppId element nested under a Fragment, Module, or Product element. To associate an AppId with a non-advertised class, nest the class within a parent AppId 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	<p>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 space-delimited list containing one or more of the following:</p> <p><i>LocalServer</i></p>
---------	------	---

A 16-bit local server application.

LocalServer32

A 32-bit local server application.

InprocServer

A 16-bit in-process server DLL.

InprocServer32

A 32-bit in-process 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	YesNoType	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	Enumeration	Threading model for the CLSID. This attribute's value must be one of the following: <i>apartment</i> <i>free</i> <i>both</i> <i>neutral</i> <i>single</i> <i>rental</i>
Version	String	Version for the CLSID.

Remarks

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

Id/Context/Server

In General

```
[HKCR\CLSID\{Id}\Context1]
@="!Server"
[HKCR\CLSID\{Id}\Context2]
@="!Server"
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}\LocalServer]
@="!comserv.dll"
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}\LocalServer32]
@="!comserv.dll"
```

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"
```

AppId

In General

```
[HKCR\CLSID\{Id}]
AppId="{AppId}"
```

Specific Example

```
[HKCR\CLSID\{01234567-89AB-CDEF-0123-
456789ABCDEF}]
AppId="{00000000-89AB-0000-0123-000000000000}"
```

Argument

In General

```
[HKCR\CLSID\{Id}\Context]
```

@="![Server] Argument"

Specific Example

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

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

Control

In General

Value "yes" specified:

[HKCR\CLSID\{Id}\Control]

Specific Example

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

Description

In General

[HKCR\CLSID\{Id}]

@="Description"

Specific Example

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

@="Description of Example COM Component"

Handler

In General

Value "1" specified:

[HKCR\CLSID\{Id}\InprocHandler]

@="ole.dll"

Value "2" specified:

[HKCR\CLSID\{Id}\InprocHandler32]

@="ole32.dll"

Value "3" specified:

[HKCR\CLSID\{Id}\InprocHandler]

@="ole.dll"

[HKCR\CLSID\{Id}\InprocHandler32]

@="ole32.dll"

Other value specified:

[HKCR\CLSID\{Id}\InprocHandler32]

@="Handler"

Specific Example (for other value)

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

Icon/IconIndex

This is not currently handled properly.

Insertable

In General

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

Specific Example

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

Programmable

In General

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

Specific Example

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

RelativePath

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

SafeForInitializing

In General

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

Specific Example

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

SafeForScripting

In General

Value "yes" specified:

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

Specific Example

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

ThreadingModel

In General

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

Specific Example

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

TypeLibId (from parent TypeLib/@Id)

In General

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

Specific Example

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

Version

In General

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

Specific Example

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

See Also

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

Version 3.0.5419.0

Column Element

Description

Column definition for a Custom Table

Windows Installer references

None

Parents

[CustomTable](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the column.	Yes
Category	Enumeration	Category of this column. This attribute 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> <i>UpperCase</i> <i>LowerCase</i> <i>Integer</i> <i>DoubleInteger</i> <i>TimeDate</i> <i>Identifier</i>	

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	YesNoType	Whether this column can be localized.
MaxValue	Integer	Maximum value for a

		numeric value, date or version in this column.
MinValue	Integer	Minimum value for a numeric value, date or version in this column.
Modularize	Enumeration	<p>How this column should be modularized, if at all. This attribute's value must be one of the following:</p> <p><i>None</i> Column should not be modularized. This is the default value.</p> <p><i>Column</i> Column should be modularized.</p> <p><i>Condition</i> Column is a condition and should be modularized.</p> <p><i>Icon</i> When the column is an primary or foreign key to the Icon table it should be modularized special.</p> <p><i>Property</i> Any Properties in the column should be modularized.</p> <p><i>SemicolonDelimited</i> Semi-colon list of keys, all of which need to be modularized.</p>
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	<p>The type of this column. This attribute's value must be one of the following:</p> <p><i>binary</i> 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.</p> <p><i>int</i> Column contains an integer or datetime value (the MinValue and MaxValue attributes should also be set).</p> <p><i>string</i> Column contains a non-localizable string value.</p>	Yes
Width	Integer	Width of this column.	

See Also

[Wix Schema](#)

ComboBox Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

ComplianceCheck Element

Description

Adds a row to the CCPSearch table.

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

- Sequence (min: 1, max: 1)
 1. [ComplianceDrive](#) (min: 0, max: 1): Starts searches from the CCP_DRIVE.
 2. [ComponentSearch](#) (min: 0, max: unbounded)
 3. [RegistrySearch](#) (min: 0, max: unbounded)
 4. [IniFileSearch](#) (min: 0, max: unbounded)
 5. [DirectorySearch](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

None

See Also

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

ComplianceDrive Element

Description

Sets the parent of a nested DirectorySearch element to CCP_DRIVE.

Windows Installer references

None

Parents

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

Inner Text

None

Children

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

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

Attributes

None

See Also

[Wix Schema](#)

Component Element

Description

Component for parent Directory

Windows Installer references

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

Parents

[ComponentGroup](#), [Directory](#), [DirectoryRef](#), [Feature](#),
[FeatureGroup](#), [FeatureRef](#), [Fragment](#), [Module](#), [Product](#)

Inner Text

None

Children

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

- [AppId](#) (min: 0, max: unbounded)
- [Category](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [Condition](#) (min: 0, max: unbounded)
- [CopyFile](#) (min: 0, max: unbounded)
- [CreateFolder](#) (min: 0, max: unbounded)
- [Environment](#) (min: 0, max: unbounded)
- [Extension](#) (min: 0, max: unbounded)
- [File](#) (min: 0, max: unbounded)
- [IniFile](#) (min: 0, max: unbounded)
- [Interface](#) (min: 0, max: unbounded)
- [IsolateComponent](#) (min: 0, max: unbounded)
- [ODBCDataSource](#) (min: 0, max: unbounded)
- [ODBCDriver](#) (min: 0, max: unbounded)
- [ODBCTranslator](#) (min: 0, max: unbounded)
- [ProgId](#) (min: 0, max: unbounded)
- [Registry](#) (min: 0, max: unbounded)
- [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)
- [ServiceConfig](#) (min: 0, max: unbounded)
- [ServiceConfigFailureActions](#) (min: 0, max: unbounded)
- [ServiceControl](#) (min: 0, max: unbounded)
- [ServiceInstall](#) (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'
- [Certificate](#)
- [ComPlusApplication](#)
- [ComPlusApplicationRole](#)
- [ComPlusAssembly](#)
- [ComPlusGroupInApplicationRole](#)
- [ComPlusGroupInPartitionRole](#)
- [ComPlusPartition](#)
- [ComPlusPartitionRole](#)
- [ComPlusPartitionUser](#)
- [ComPlusRoleForComponent](#)
- [ComPlusRoleForInterface](#)
- [ComPlusRoleForMethod](#)
- [ComPlusSubscription](#)
- [ComPlusUserInApplicationRole](#)
- [ComPlusUserInPartitionRole](#)
- [Driver](#)
- [EventSource](#)
- [FileShare](#)
- [FirewallException](#)

- [InternetShortcut](#)
- [MessageQueue](#)
- [MessageQueuePermission](#)
- [PerformanceCategory](#)
- [ServiceConfig](#)
- [SqlDatabase](#)
- [SqlScript](#)
- [SqlString](#)
- [User](#)
- [WebAppPool](#)
- [WebDir](#)
- [WebFilter](#)
- [WebProperty](#)
- [WebServiceExtension](#)
- [WebSite](#)
- [WebVirtualDir](#)
- [XmlConfig](#)
- [XmlFile](#)

Attributes

Name	Type	Description
Id	String	Component identifier; is the primary key for identifying component.
ComPlusFlags	Integer	Set this attribute to create a ComPlus entry. The value should be the export flags used during the generation of the .idl file. For more information see the COM+ documentation in the Platform SDK.
Directory	String	Sets the Directory of the Component. If this

		<p>element is nested under a Directory element, the value defaults to the value of the parent Directory/@Id.</p>
DisableRegistryReflection	YesNoType	<p>Set this attribute to 'yes' in order to disable registry reflection on an existing and new registry 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 Windows Installer version 4.0 and is ignored on 32-bit systems.</p>
DiskId	Integer	<p>This attribute provides the default DiskId attribute for all child File elements. Specifying the DiskId on a Component element will override any DiskId attributes set by parent Directory or DirectoryFile elements. See the File element's DiskId attribute for more information about the purpose of the DiskId.</p>
Feature	String	<p>Identifies a feature to which this component belongs, as a shorthand for a child Component</p>

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 attribute allows you to specify a single feature.

Guid	ComponentGuid	This value should be a guid that uniquely identifies this component's contents, language, platform, and version. It's also possible to set the value to an empty string to specify unmanaged components. Unmanaged components are a security vulnerability because the component cannot be removed or repaired by Windows Installer (it is essentially an unpatchable, permanent component). Therefore, the guid should always be specified for any component which contains resources that may need to be patched in the future.
KeyPath	YesNoType	If this attribute's value is 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 'y' on one of those child elements.
Location	Enumeration	<p>Optional value that specifies the location that the component can be run from. This attribute value must be one of the following:</p> <p><i>local</i> Prevents the component from running from the source or the network (this is the default behavior if this attribute is not set).</p> <p><i>source</i> Enforces that the component can only be run from the source (it cannot be run from the user's computer).</p> <p><i>either</i> Allows the component to run from source or locally.</p>
NeverOverwrite	YesNoType	If this attribute is set to

'yes', the installer does not install or reinstall the component if a key path registry entry for the component already exists. The application does register itself as a client of the component. Use this flag only for components that are being registered by the Registry table. Do not use this flag for components registered by the AppId, Class, Extension, ProgId, MIM 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 system client for the component in the Windows Installer registry settings (which basically just means that at least one product is always referencing this component). Note that this option differs from the behavior of not setting a GUID because although the component is permanent, it is still patchable (because Windows Installer still tracks it), it's just not

Shared	YesNoType	<p>uninstallable.</p> <p>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.</p>
SharedDllRefCount	YesNoType	<p>If this attribute's value set to 'yes', the installer increments the referer count in the shared DL registry of the component's key file. If this bit is not set, the installer increments the reference count only if the reference count already exists.</p>
Transitive	YesNoType	<p>If this attribute is set to 'yes', the installer reevaluates the value the statement in the Condition upon a reinstall. If the value was previously False and has changed to True, the installer installs the component. If the value was previously True and has changed to False, the installer removes t</p>

		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	YesNoType	Set this attribute to 'yes' to mark this as a 64-bit component. This attribute facilitates the installation of packages that include both 32-bit and 64-bit components. If this attribute is not set, the component is registered as a 32-bit component. If this is a 64-bit component replacing a 32-bit component, set this attribute to 'yes' and assign a new GUID in the Guid attribute.

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

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'](#)

Attributes

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

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

Attributes

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

under only one feature, that feature is the primary feature for the component. You cannot set more than one feature as the primary feature of a given component.

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

See Also

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

Version 3.0.5419.0

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

Attributes

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

features. If a component nests under only one feature, that feature is the primary feature for the component. You cannot set more than one feature as the primary feature of a given component.

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

How Tos and Examples

- [How To: Add a file to your installer](#)

See Also

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

Version 3.0.5419.0

ComponentSearch Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	String		Yes
Guid	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: <i>directory</i> 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](#)

Version 3.0.5419.0

Condition Element

Description

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

Windows Installer references

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

Parents

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

Attributes

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

Attributes

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

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

See Also

[Wix Schema](#)

Control Element

Description

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

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

Attributes

Name	Type	Description	Re
Id	String	Combined with the Dialog Id to make up the primary key of the Control table.	Yes
Bitmap	YesNoType	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	YesNoType	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	YesNoType	This attribute is only valid for Volume and Directory Controls.	
FormatSize	YesNoType	This attribute is only valid for Text Controls.	
HasBorder	YesNoType	This attribute is only valid for RadioButton Controls.	
Height	LocalizableInteger	Height of the rectangular boundary of the control. This must be a non-negative	Yes

		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	YesNoType	This attribute is only valid for Edit Controls.
NoPrefix	YesNoType	This attribute is only valid for Text Controls.
NoWrap	YesNoType	This attribute is only valid for Text Controls.
Password	YesNoType	This attribute is only valid for Edit Controls.
ProgressBlocks	YesNoType	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	YesNoType	This attribute is only valid for RadioButton and Checkbox Controls.
RAMDisk	YesNoType	This attribute is only valid for Volume and Directory Controls.
Remote	YesNoType	This attribute is only valid for Volume and Directory Controls.
Removable	YesNoType	This attribute is only valid for Volume and Directory Controls.
RightAligned	YesNoType	Set this attribute to "yes" to cause the Control to be right aligned.
RightToLeft	YesNoType	Set this attribute to "yes" to cause the Control to display from right to left.
ShowRollbackCost	YesNoType	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	YesNoType	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	Yes
UserLanguage	YesNoType	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.	Yes
X	LocalizableInteger	Horizontal coordinate of the upper-left corner of the rectangular boundary of the control. This must be	Yes

		a non-negative number.
Y	LocalizableInteger	Vertical coordinate of the upper-left corner of the rectangular boundary of the control. This must be a non-negative number.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

Attributes

Name	Type	Description
Id	String	Primary key used to identify this parent entry.

Delete	YesNoType	This attribute be specified if element is ne: under a File e or the FileId a is specified. In cases, if the a is not specifie default value i and the file is not moved. Set value to "yes" to move the fi deleting the s file) instead of it.
DestinationDirectory	String	Set this value destination dir where an exists on the target i should be mo copied to. This Directory must the installer d at creation tim attribute can specified in conjunction w DestinationPr
DestinationLongName	LongFileNameType	This attribute been depreca please use the DestinationNa attribute inste
DestinationName	LongFileNameType	In prior versio WiX toolset, th attribute spec short file nam

set this value localizable name given to the object file after it is not copied. If this is not specified the destination given the same as the source short file name specified, the DestinationShort attribute may be specified. If a name is specified DestinationLocal attribute may be specified. Also value is a long name, the DestinationShort attribute may be omitted to allow to attempt to generate a unique short name. However name collides with another file or wish to manually specify the short name, then the DestinationShort attribute may be specified.

DestinationProperty	String	Set this value property that is a value that refers to the full path
---------------------	--------	--

		destination dir The property must have to exist in installer database creation time; be created at installation time custom action command line This attribute be specified in conjunction with DestinationDir
DestinationShortName	ShortFileNameType	The short file the file in 8.3.1 This attribute only be set if it conflict between generated short names or you manually specify short file name
FileId	String	This attribute be specified if element is needed under a File element Set this attribute value to the id of a file from a different component copy it based install state of parent component
SourceDirectory	String	This attribute be specified if element is needed under a File element or the FileId attribute

is specified. S value to the s directory from copy or move existing file or target machin Directory mus the installer d: at creation tim attribute 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 l removed from specified direc 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 (anc element is not

under a File element or specify a File attribute) then SourceProperty attribute should resolve to the name of the source filename. If the value of this attribute contains a "*" and the DestinationName attribute is specified, all moved or copied files retain the names from the sources.

SourceProperty	String	This attribute can be specified if the element is nested under a File element or the FileId attribute is specified. Set the value to a property that will have a path that resolves to the path of the source directory (or file, including file name if SourceName is specified). The property does not have to exist in the installer database at creation time; it can be created at
----------------	--------	---

installation time
custom action
command line
This attribute
be specified in
conjunction with
SourceDirectory

See Also

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

Version 3.0.5419.0

CostFinalize Element

Description

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

Windows Installer references

[CostFinalize Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

sequence.

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

See Also

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

Version 3.0.5419.0

CostInitialize Element

Description

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

Windows Installer references

[CostInitialize Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

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

CreateFolder Element

Description

Create folder as part of parent Component.

Windows Installer references

[CreateFolder Table](#)

Parents

[Component](#)

Inner Text

None

Children

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

- [Permission](#) (min: 0, max: unbounded): ACL permission
- [PermissionEx](#) (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'](#)
- [PermissionEx](#)

Attributes

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

See Also

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

CreateFolders Element

Description

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

Windows Installer references

[CreateFolders Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

CreateShortcuts Element

Description

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

Windows Installer references

[CreateShortcuts Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Custom Element

Description

Use to sequence a custom action.

Windows Installer references

None

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
Action	String	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	Enumeration	Mutually exclusive with	

Before, After, and Sequence attributes This attribute's value must be one of the following:

success

cancel

error

suspend

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 [Custom Action Types](#) 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

Attributes

Name	Type	Description
BinaryKey	String	This attribute is a reference to a key in the Product table with matching Id attribute. The key contains the custom action filename. The custom action will not be installed in the target directory. This attribute is used with the DllEntry attribute to specify the action DLL to use for a type of custom action with the ExeCommand attribute. This attribute is used with the 17 custom action that runs a script, or with the VbsScriptCall or JScriptCall attributes to specify the script to run a custom action.
Directory	String	This attribute specifies a reference to a directory in the Product table.

element with matching Id attribute and directory path. This attribute is used with the ExeCommand attribute to specify the source executable for a type 1 custom action or with the Value attribute to specify a string to place in the specified entry in a type 35 custom action.

DllEntry	String	This attribute specifies the name of a custom action to execute. It is used with the BinaryKey attribute for a type 1 custom action, or with the Value attribute to create a type 17 custom action.
Error	String	This attribute specifies an integer value to use as an error message ID for a custom action that displays a message box and aborts a product's installation.
ExeCommand	String	This attribute specifies the command-line parameters to supply to an executable. This attribute is used with the BinaryKey attribute for a type 1 custom action, the FileKey attribute for a type 2 custom action, the Property attribute for a type 3 custom action, or the Directory attribute for a type 34 custom action that specifies the path to an executable to run.
Execute	Enumeration	This attribute indicates the execution mode for a custom action. This attribute can be one of the following: <i>commit</i> Indicates that the custom action runs after successful completion of the installation script (at the end of the installation). <i>deferred</i> Indicates that the custom action runs after the installation script (possibly with elevated privileges). <i>firstSequence</i>

Indicates that the custom action is executed in the first sequence that is not deferred.

immediate

Indicates that the custom action is executed during normal processing of the installation. This is the default.

oncePerProcess

Indicates that the custom action is executed in the first sequence that is not deferred for each process.

rollback

Indicates that a custom rollback sequence will be executed during installation, usually after a custom action made by a deferred custom action.

secondSequence

Indicates that a custom action will be executed a second time if it was previously executed in an earlier sequence.

FileKey	String	This attribute specifies a reference to a file element with matching Id attribute. The custom action will execute the custom action command specified in the ExeCommand attribute of the file element when the file is installed. This attribute is used in conjunction with the DllEntry attribute of the 18 custom action that runs a custom action that runs an executable, with the DllEntry attribute of the 17 custom action, or with the JScriptCall attributes to specify the command to execute a custom action.
HideTarget	YesNoType	Ensures the installer does not display the custom action data for the deferred custom action.
Id	String	The identifier of the custom action.
Impersonate	YesNoType	This attribute specifies whether the custom action should impersonate the user specified in the Impersonate attribute.

		Installer, which executes as should impersonate the user installing user when executing action. Typically the value should be the user name except when the custom action requires elevated privileges to apply changes to the system.
JScriptCall	String	This attribute specifies the name of the JavaScript function to execute in a script. The script can be provided in a Binary element. The BinaryKey attribute describes the script file. In other words, this attribute must be used in conjunction with the BinaryKey attribute.
PatchUninstall	YesNoType	This attribute specifies that the custom patch is being uninstalled. The custom action should also be conditioned with the MSIPATCHREMOVE property. The property should be set to a lower level (less than Windows Installer) behavior.
Property	String	This attribute specifies a reference to a property element with matching Id attribute. The Property attribute is used to specify the Property to be used or updated by this custom action. This attribute is used with the Value attribute to specify the value of this custom action that parses the property. The Property attribute places it into the specified Property element. The Property attribute is also used with the Value attribute to create a type 50 property. The Property attribute uses the value of the given property to specify the path to the executable. The Property attribute is often useful to parse a deferred custom action. See http://msdn.microsoft.com/lik for more information.
Return	Enumeration	Set this attribute to set the return value of the custom action. This attribute can be one of the following: <i>asyncNoWait</i>

Indicates that the custom action will execute asynchronously and the return code will be checked after the installer terminates.

asyncWait

Indicates that the custom action will execute asynchronously but the return code will be checked at sequence completion.

check

Indicates that the custom action will execute synchronously and the return code will be checked for success. The return code will be checked only if the action is successful.

ignore

Indicates that the custom action will execute synchronously and the return code will not be checked.

Script	Enumeration	Creates a type 37 or 38 custom action. The value of the element should contain the name of the script embedded in the package. The value must be one of the following: <i>jscript</i> <i>vbscript</i>
SuppressModularization	YesNoType	Use to suppress modularization of a custom action name in merge modules. This is necessary for table-driven actions because the table name which is used cannot be modularized, so there will be only one instance of the table.
TerminalServerAware	YesNoType	This attribute specifies whether a custom action will impersonate the local user during per-machine installation on Terminal Server machines. Deferred custom actions that do not specify this attribute explicitly set it 'no', will run without impersonation on Terminal Server machines during per-machine installation.

		only applicable when installing Server 2003 family.
Value	String	This attribute specifies a string for the custom action. This attribute is used in conjunction with the Property attribute to set a value for a part of a type 51 custom action. This attribute is used in conjunction with the Directory attribute to set a directory path. This attribute can be a literal value or derived from an element using the FormattedText attribute.
VBScriptCall	String	This attribute specifies the name of the Subroutine to execute in a script. The script must be provided in a Binary key by the BinaryKey attribute. In other words, this attribute must be used in conjunction with the BinaryKey attribute.
Win64	YesNoType	Specifies that a script custom action is for a 64-bit platform. Valid only with the Script, VBScriptCall, and JavaScript attributes.
Any attribute namespace='###other' processContents='lax'		

See Also

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

Version 3.0.5419.0

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

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

See Also

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

CustomProperty Element

Description

A custom property for the PatchMetadata table.

Windows Installer references

None

Parents

[PatchMetadata](#)

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

CustomTable Element

Description

Defines a custom table for use from a custom action.

Windows Installer references

None

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

Data Element

Description

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

See Also

[Wix Schema](#)

Dependency Element

Description

Declares a dependency on another merge module.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

Dialog Element

Description

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

Attributes

Name	Type	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.	
Y	Integer	Vertical placement of the dialog box as a percentage of screen height. The default value is 50.	

See Also

[Wix Schema](#)

Version 3.0.5419.0

DialogRef Element

Description

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

Windows Installer references

None

Parents

[UI](#)

Inner Text

None

Children

None

Attributes

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

See Also

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

DigitalCertificate Element

Description

Adds a digital certificate.

Windows Installer references

[MsiDigitalCertificate Table](#)

Parents

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

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

DigitalSignature Element

Description

Adds a digital signature.

Windows Installer references

[MsiDigitalSignature Table](#)

Parents

[Media](#)

Inner Text

None

Children

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

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

Attributes

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

See Also

[Wix Schema](#)

Directory Element

Description

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

Windows Installer references

[Directory Table](#)

Parents

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

Inner Text

None

Children

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

- [Component](#) (min: 0, max: unbounded)
- [Directory](#) (min: 0, max: unbounded)
- [Merge](#) (min: 0, max: unbounded)
- [SymbolPath](#) (min: 0, max: unbounded)
- [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description
Id	String	This value is identifier of the
ComponentGuidGenerationSeed	Guid	The Component Generation Seed must be used with Component guid directive rooted in a source Installer directory. For example, Product or Component

is recommended that the `Directory` attribute be assigned by developers in the `Components` directories with the `Directory` attribute instead (for example, `ProgramFile` Name Product Version"). It is noted that once assigned a `Component` Generation `Source` must not change the directory name.

DiskId	Integer	Sets the default directory for the files component. This may be overridden by the <code>Component</code> , <code>File</code> element, or <code>File</code> element. Merge element attribute for <code>File</code> .
FileSource	String	Used to set the source for the elements. For information, see source files .
LongName	LongFileNameType	This attribute is deprecated; use <code>Name</code> attribute.
LongSource	LongFileNameType	This attribute is deprecated; use <code>SourceName</code> attribute.
Name	LongFileNameType	The name of the directory.

Do not specify the `LongName` attribute for the directory representation.

directory as the Windows [Directory tab](#) information operator).

In prior versions of the toolset, this attribute's value was either a short name. If a short name is specified, ShortName can be specified. If a long name is specified, LongName can be specified. If a long directory name is specified, ShortName can be omitted to all attempts to get short directory information. However, if the directory is specified with another name, you can wish to manually specify short directory information. ShortName can be specified.

ShortName	ShortFileNameType	The short name of the directory in the file system. If the attribute is present, there is a conflict between the generated short names or the manually specified directory name.
ShortSourceName	ShortFileNameType	The short name of the source directory.

directory on 1
in 8.3 format
should only k
conflict betw
short directo
user wants to
the short sou
name.

SourceName

[LongFileNameType](#)

The name of
the source r
attribute is n
Windows Ins
to the Name

In prior versi
toolset, this a
the short sou
name. This a
may now be
long director
directory nar
the ShortSou
attribute may
If a long dire
specified, the
attribute may
Also, if this v
directory nar
ShortSource
may be omit
to attempt to
unique short
However, if t
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](#)

Version 3.0.5419.0

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'](#)

Attributes

Name	Type	Description	Required
Id	String	The identifier of the Directory element to reference.	Yes
DiskId	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'

How Tos and Examples

- [How To: Add a file to your installer](#)

See Also

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

Version 3.0.5419.0

DirectorySearch Element

Description

Searches for directory and assigns to value of parent Property.

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	String	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)

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

Attributes

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

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

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](#)

Version 3.0.5419.0

DisableRollback Element

Description

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

Windows Installer references

[DisableRollback Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

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	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](#)

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

Attributes

Name	Type	Description	Required
Id	String	Unique identifier for embedded chainer.	Yes
BinarySource	String	Reference to the Binary element that contains the chainer executable. 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 executable. Mutually exclusive with the BinarySource and PropertySource attributes.
PropertySource	String	Reference to a Property that resolves to the full path to the chainer executable. Mutually exclusive with the BinarySource and FileSource attributes.

See Also

[Wix Schema](#), [Binary](#), [File](#), [Property](#), [EmbeddedChainerRef](#)

Version 3.0.5419.0

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	Type	Description	Required
Id	String		Yes
Any attribute namespace='##other' processContents='lax'			

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. [EmbeddedUIResource](#) (min: 0, max: unbounded): Specifies extra files to be extracted for use by the embedded UI, such as language resources.

Attributes

Name	Type	Description
Id	String	Unique identifier for an attribute is not specified attribute or the filename. SourceFile attribute will
IgnoreActionData	YesNoType	Embedded UI will not record INSTALLLOGMODE_/_ messages.
IgnoreActionStart	YesNoType	Embedded UI will not record INSTALLLOGMODE_/_ messages.
IgnoreCommonData	YesNoType	Embedded UI will not record 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	YesNoType	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_C messages.
IgnoreProgress	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreResolveSource	YesNoType	Embedded UI will not r INSTALLLOGMODE_F messages.
IgnoreRMFilesInUse	YesNoType	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_T

		messages.
IgnoreUser	YesNoType	Embedded UI will not r INSTALLLOGMODE_U
IgnoreWarning	YesNoType	Embedded UI will not r INSTALLLOGMODE_\n messages.
Name	LongFileNameType	The name for the emb it is extracted from the executed. (Windows Ir support the typical sho filename combination f files as it does for othe this attribute is not spe portion of the SourceF used.
SourceFile	String	Path to the binary file t embedded UI. This mu exports the following th InitializeEmbeddedUI, EmbeddedUIHandler a ShutdownEmbeddedU
SupportBasicUI	YesNoType	Set yes to allow the W display the embedded level installation.

See Also

[Wix Schema](#)

EmbeddedUIResource Element

Description

Defines a resource for use by the embedded UI.

Windows Installer references

[MsiEmbeddedUI Table](#)

Parents

[EmbeddedUI](#)

Inner Text

None

Children

None

Attributes

Name	Type	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	Yes

	specified the Id attribute will be used.	
SourceFile String	Path to the binary file that is the embedded UI resource.	Yes

See Also

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

Version 3.0.5419.0

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

Remarks

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

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

See Also

[Wix Schema](#)

Version 3.0.5419.0

Environment Element

Description

Environment variables added or removed for the parent component.

Windows Installer references

[Environment Table](#)

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

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

it already exists.

set

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

remove

Removes the environment variable during an installation. The installer only removes an environment variable during an installation if the name and value of the variable match the entries in the Name and Value attributes. If you want to remove an environment variable, regardless of its value, do not set the Value attribute.

Name	String	Name of the environment variable.	Yes
Part	Enumeration	This attribute's value must be one of the following: <i>all</i> This value is the entire environmental variable. This is the default. <i>first</i>	

This value is prefixed.

last

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

See Also

[Wix Schema](#)

Exclusion Element

Description

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

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

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

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

See Also

[Wix Schema](#)

ExecuteAction Element

Description

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

Windows Installer references

[ExecuteAction Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Extension Element

Description

Extension for a Component

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description
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'
IsRichSavedGame		Registers this extension for the rich property handler on Windows Vista (http://schemas.microsoft.com/wix/)

See Also

[Wix Schema](#)

ExternalFile Element

Description

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

Windows Installer references

None

Parents

[Family](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

Failure Element

Description

Failure action for a ServiceConfigFailureActions element.

Windows Installer references

None

Parents

[ServiceConfigFailureActions](#)

Inner Text

None

Children

None

Attributes

Name	Type	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

[Wix Schema](#)

Family Element

Description

Group of one or more upgraded images of a product.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

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

See Also

[Wix Schema](#)

Version 3.0.5419.0

Feature Element

Description

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

Windows Installer references

[Feature Table](#)

Parents

[Feature](#), [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'](#)

Attributes

Name	Type	Description
Id	String	Unique identifier of the feature.
Absent	Enumeration	This attribute determines if a user option to set a feature to absent. This attribute's value must be <i>allow</i> . Allows the user interface to

change the feature state to

disallow

Prevents the user interface option to change the feature setting the `msidbFeatureAttributesUI` attribute. This will force the installation state, whether visible in the UI.

AllowAdvertise	Enumeration	This attribute determines the possible states for this feature. This attribute can be one of the following: <i>no</i> Prevents this feature from setting the <code>msidbFeatureAttributesDisallow</code> attribute. <i>system</i> Prevents advertising for the operating system shell documents. Windows Installer description file sets the <code>msidbFeatureAttributesNoShell</code> attribute. <i>yes</i> Allows the feature to be advertised.
ConfigurableDirectory	String	Specify the Id of a Directory that can be set by the user at installation time. This attribute must be a public property and therefore its name must be uppercase.
Description	String	Longer string of text describing the feature. A localized string is displayed by the Selection Dialog.
Display	String	Determines the initial display order of the feature tree. This attribute's value can be one of the following:

collapse

Initially shows the feature in its default value.

expand

Initially shows the feature in its expanded state.

hidden

Prevents the feature from appearing in the interface.

<an explicit integer value>

For advanced users only, it allows you to set the integer value of the `msidbFeatureAttribute` to appear in the Feature row.

InstallDefault	Enumeration	This attribute determines the default location of a feature. This attribute is specified if the value of the <code>FOLIO</code> is 'yes' since that would ask the installer to follow the parent installation simultaneously favor a particular feature just for this feature. This attribute can be one of the following: <i>followParent</i> Forces the feature to follow the state as its parent feature. <i>local</i> Favors installing this feature locally. This is the <code>msidbFeatureAttributeSource</code> . <i>source</i> Favors running this feature from the source by setting the <code>msidbFeatureAttributeSource</code> attribute.
Level	Integer	Sets the install level of this feature. 0 is to disable the feature. Processing can modify the level value (this is a Condition child element).

Title	String	Short string of text identifying the feature. This attribute is listed as an item by the Selection Dialog.
TypicalDefault	Enumeration	This attribute determines the default installation option of the feature. This attribute's value can be one of the following: <i>advertise</i> Sets the feature to be advertised. This value cannot be set if the AllowAdvertise attribute is set to false. This value asks the installer to disallow installation for this feature while at the user's discretion. <i>install</i> Sets the feature to the default installation option.

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

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)
- [ComponentGroupRef](#) (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'](#)

Attributes

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

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

Attributes

Name	Type	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	YesNoType	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'

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)
- [ComponentGroupRef](#) (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'](#)

Attributes

Name	Type	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'

See Also

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

Version 3.0.5419.0

File Element

Description

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

Windows Installer references

[File Table](#)

Parents

[Component](#)

Inner Text

None

Children

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

- [AppId](#) (min: 0, max: unbounded)
- [AssemblyName](#) (min: 0, max: unbounded)
- [Class](#) (min: 0, max: unbounded)
- [CopyFile](#) (min: 0, max: unbounded): Used to create a duplicate of this file elsewhere.
- [ODBCDriver](#) (min: 0, max: unbounded)
- [ODBCTranslator](#) (min: 0, max: unbounded)
- [Permission](#) (min: 0, max: unbounded): Used to configure the ACLs for this file.
- [PermissionEx](#) (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'](#)
- [EventManifest](#)
- [FirewallException](#)

- [FormatsFile](#)
- [Game](#)
- [HelpCollection](#)
- [HelpFile](#)
- [NativeImage](#)
- [PerfCounter](#)
- [PerfCounterManifest](#)
- [PermissionEx](#)
- [SnapIn](#)
- [TypesFile](#)

Attributes

Name	Type	Description
Assembly	Enumeration	<p>Specifies if this File Win32 Assembly or .NET Assembly that needs to be installed into the Global Assembly Cache (GAC). If the value is '.net' or 'win32', this file must also be the key path for the Component. This attribute's value must be one of the following:</p> <p><i>.net</i> The file is a .NET Framework assembly.</p> <p><i>no</i> The file is not a .NET Framework Win32 assembly. This is the default value.</p> <p><i>win32</i></p>

The file is a Win 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 this attribute is absent, the assembly will be installed to the Global Assembly Cache (GAC). This attribute may only be specified if the AssemblyAttribute is set to '.net' or 'win32'.
AssemblyManifest	String	Specifies the file identifier of the manifest file that describes the assembly. The manifest file should be in the same component as the assembly it describes. This attribute may only be specified if the AssemblyAttribute is set to '.net' or 'win32'.
BindPath	String	A list of paths, separated by semicolons, that represent the paths to be searched to find imported DLLs. The list is usually a list of properties, with each property enclosed in square brackets. The

value may be set to empty string. Including this attribute will cause an entry to be generated for the file in the BindImage table.

Checksum	YesNoType	This attribute should be set to "yes" for every executable file in the installation that has a valid checksum stored in the Portable Executable (PE) file header. On those files that have this attribute set will be verified for a valid checksum during a reinstall.
CompanionFile	String	Set this attribute to make this file a companion child of another file. The installation state of a companion file depends not on its own file versioning information but on the versioning information of its companion parent file that is the key parent file for its component cannot be a companion (that means this attribute cannot be set to KeyPath="yes" for this file). The Version attribute cannot be set along with this attribute since companion file

are not installed bas
on their own versior

Compressed	YesNoDefaultType	Sets the file's source type compression. A 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 suppre files option is not us
DefaultSize	Integer	This is the default si of this file. The linke replace this value fr the value in the file i 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 suppre files option is not us
DiskId	Integer	The value of this attribute should correspond to the Id attribute of a Media element authored elsewhere. By creat this connection betw a file and its media, set the packaging options to the value: specified in the Med element (values suc

compression level, (embedding, etc...).

Specifying the DiskId attribute on the File element overrides the default DiskId attribute from the parent Component element. If no DiskId attribute is specified, the default "1". This DiskId attribute is ignored when creating a merge module because merge modules do not have media.

FontTitle	String	Causes an entry to be generated for the file in the Font table with the specified FontTitle. This attribute is intended to be used to register the file as a non-TrueType font.
Hidden	YesNoType	Set to yes in order to have the file's hidden attribute set when it is installed on the target machine.
Id	String	The unique identifier for this File element. If you omit Id, it defaults to the file name portion of the Source attribute, if specified. May be referenced as a Property by specifying [#value].
KeyPath	YesNoType	Set to yes in order to

force this file to be the key path for the parent component.

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 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. 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 this name collides with another or you wish to manually specify the short file name, then the ShortName attribute may be specified. Finally, if this attribute is omitted then its default value is the file name portion of the Source

attribute, if one is specified, or the value of the Id attribute, if the Source attribute is omitted or doesn't contain a file name.

PatchAllowIgnoreOnError	YesNoType	Set to indicate that the patch is non-vital.
PatchGroup	Integer	This attribute must be set for patch-added files. Each patch should be assigned a different patch group number. Patch groups numbers must be greater than 0 and should be assigned consecutively. For example, the first patch should use PatchGroup='1', the second patch will have PatchGroup='2', etc.
PatchIgnore	YesNoType	Prevents the updating of the file that is in fact changed in the upgraded image relative to the target images.
PatchWholeFile	YesNoType	Set if the entire file should be installed rather than creating a binary patch.
ProcessorArchitecture	Enumeration	Specifies the architecture for this assembly. This attribute should only be used for .NET Framework 2.0 and higher assemblies. The attribute's value must

one of the following:

msil

The file is a .NET Framework assembly that is processor-neutral.

x86

The file is a .NET Framework assembly for the x86 processor.

x64

The file is a .NET Framework assembly for the x64 processor.

ia64

The file is a .NET Framework assembly for the ia64 processor.

ReadOnly	YesNoType	Set to yes in order to have the file's read-only attribute set when it is installed on the target machine.
SelfRegCost	Integer	The cost of registering the file in bytes. This must be a non-negative number. Including this attribute will cause an entry to be generated for the file in the SelfReg table.
ShortName	ShortFileNameType	The short file name of the file in 8.3 format.

This attribute should only be set if there is a conflict between generated short file names or the user wants to manually specify a short file name.

Source	String	Specifies the path to the file in the build process. Overrides default source path set by parent directories and Name attribute. This attribute must be set if no source information can be gathered from parent directories. For more information, see Specifying source file .
src	String	This attribute has been deprecated; please use the Source attribute instead.
System	YesNoType	Set to yes in order to have the file's system attribute set when it is installed on the target machine.
TrueType	YesNoType	Causes an entry to be generated for the file in the Font table with the FontTitle specified. This attribute is intended to be used to register the file as a TrueType font.
Vital	YesNoType	If a file is vital, then installation cannot proceed unless the

is successfully installed. The user will have the option to ignore an error installing this file. If an error occurs, they can merely retry to install the file or abort the installation. The default is "yes," unless the `sfdvital` switch (candle.exe) or `SuppressFileDefault` property (.wixproj) is used.

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

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

Windows Installer references

[FileCost Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

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

Version 3.0.5419.0

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

Attributes

Name	Type	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	LongFileNameType	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	ShortFileNameType	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

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

Version 3.0.5419.0

FileSearchRef Element

Description

References an existing FileSearch element.

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

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

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

See Also

[Wix Schema](#)

FindRelatedProducts Element

Description

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

Windows Installer references

[FindRelatedProducts Action](#)

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	

Before	String	The name of an action that this action should come before.
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](#), [Upgrade](#)

Version 3.0.5419.0

ForceReboot Element

Description

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

Windows Installer references

[ForceReboot Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

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)

- [AppId](#) (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)
- [CustomActionRef](#) (min: 0, max: unbounded)
- [CustomTable](#) (min: 0, max: unbounded)
- [Directory](#) (min: 0, max: unbounded)

- [DirectoryRef](#) (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)
- [IgnoreModularization](#) (min: 0, max: unbounded)
- [Media](#) (min: 0, max: unbounded)
- [PackageCertificates](#) (min: 0, max: unbounded)
- [PatchCertificates](#) (min: 0, max: unbounded)
- [PatchFamily](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SetDirectory](#) (min: 0, max: unbounded)
- [SetProperty](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (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. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- Any Element namespace='##other' processContents='Lax'
- [CloseApplication](#)
- [ComPlusApplication](#)
- [ComPlusApplicationRole](#)
- [ComPlusPartition](#)

- [ComPlusPartitionRole](#)
- [Group](#)
- [HelpCollectionRef](#)
- [HelpFilter](#)
- [SqlDatabase](#)
- [User](#)
- [WebApplication](#)
- [WebAppPool](#)
- [WebDirProperties](#)
- [WebLog](#)
- [WebSite](#)

Attributes

Name	Type	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, ProgId, or Class elements (but not UI controls)

Windows Installer references

[Icon Table](#)

Parents

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

Inner Text

None

Children

None

Attributes

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

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

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

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	Type	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> <i>Property</i> <i>Directory</i>	

See Also

[Wix Schema](#)

IgnoreRange Element

Description

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

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

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	Type	Description	Required
Id	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	Type	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: <i>addLine</i> Creates or updates an .ini entry. <i>addTag</i> Creates a new entry or appends a new comma-separated value to an existing	Yes

entry.

createLine

Creates an .ini entry only if the entry does not already exist.

removeLine

Removes an .ini entry.

removeTag

Removes a tag from an .ini entry.

Directory	String	Name of a property, the value of which is the full path of the folder containing the .ini file. Can be name of a directory in the Directory table, a property set by the AppSearch table, or any other property representing a full path.	
Key	String	The localizable .ini file key within the section.	Yes
LongName	LongFileNameType	This attribute has been deprecated; please use the Name attribute instead.	
Name	LongFileNameType	In prior versions of the WiX toolset, this	Yes

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 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](#)

Version 3.0.5419.0

IniFileSearch Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	String	External key into the Signature table.	Yes
Field	Integer	The field in the .ini line. If field is Null or 0, the entire line is read.	
Key	String	The key value within the section.	Yes
LongName	LongFileNameType	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	

Type	Enumeration	short name. 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: <i>directory</i> A directory location. <i>file</i> A file location. This is the default value. <i>raw</i> 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	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

InstallExecute Element

Description

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

Windows Installer references

[InstallExecute Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

InstallExecuteAgain Element

Description

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

Windows Installer references

[InstallExecuteAgain Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

InstallExecuteSequence Element

Description

None

Windows Installer references

[InstallExecuteSequence Table](#)

Parents

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

Inner Text

None

Children

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

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

removes its registration from the system.

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

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

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

- [RegisterProduct](#) (min: 0, max: unbounded): Registers the product information with the installer.
- [RegisterProgIdInfo](#) (min: 0, max: unbounded): Manages the registration of OLE ProgId information with the system.
- [RegisterTypeLibraries](#) (min: 0, max: unbounded): Registers type libraries with the system.
- [RegisterUser](#) (min: 0, max: unbounded): Registers the user information with the installer to identify the user of a product.
- [RemoveDuplicateFiles](#) (min: 0, max: unbounded): Deletes files installed by the DuplicateFiles action.
- [RemoveEnvironmentStrings](#) (min: 0, max: unbounded): Modifies the values of environment variables.
- [RemoveExistingProducts](#) (min: 0, max: unbounded): Goes through the product codes listed in the ActionProperty column of the Upgrade table and removes the products in sequence.
- [RemoveFiles](#) (min: 0, max: unbounded): Removes files previously installed by the InstallFiles action.
- [RemoveFolders](#) (min: 0, max: unbounded): Removes any folders linked to components set to be removed or run from source.
- [RemoveIniValues](#) (min: 0, max: unbounded): Removes .ini file information specified for removal in the RemoveIniFile table if the component is set to be installed locally or run from source.
- [RemoveODBC](#) (min: 0, max: unbounded): Removes the data sources, translators, and drivers listed for removal during the installation.
- [RemoveRegistryValues](#) (min: 0, max: unbounded): Removes a registry value that has been authored into the registry table if the associated component was installed locally or as run from source, and is now set to be uninstalled.
- [RemoveShortcuts](#) (min: 0, max: unbounded): Manages the removal of an advertised shortcut whose feature is selected for uninstallation or a nonadvertised shortcut whose component is selected for uninstallation.
- [ResolveSource](#) (min: 0, max: unbounded): Determines the location of the source and sets the SourceDir property if the

source has not been resolved yet. Not fixed sequence.

- [RMCCPSearch](#) (min: 0, max: unbounded): Uses file signatures to validate that qualifying products are installed on a system before an upgrade installation is performed.
- [ScheduleReboot](#) (min: 0, max: unbounded): Prompts the user to restart the system at the end of installation. Not fixed sequence.
- [SelfRegModules](#) (min: 0, max: unbounded): Processes all modules listed in the SelfReg table and registers all installed modules with the system.
- [SelfUnregModules](#) (min: 0, max: unbounded): Unregisters all modules listed in the SelfReg table that are scheduled to be uninstalled.
- [SetODBCFolders](#) (min: 0, max: unbounded): Checks for existing ODBC drivers and sets the target directory for each new driver to the location of an existing driver.
- [StartServices](#) (min: 0, max: unbounded): Starts system services.
- [StopServices](#) (min: 0, max: unbounded): Stops system services.
- [UnpublishComponents](#) (min: 0, max: unbounded): Manages the unadvertisement of components listed in the PublishComponent table.
- [UnpublishFeatures](#) (min: 0, max: unbounded): Removes selection-state and feature-component mapping information from the registry.
- [UnregisterClassInfo](#) (min: 0, max: unbounded): Manages the removal of COM class information from the system registry.
- [UnregisterComPlus](#) (min: 0, max: unbounded): Removes COM+ applications from the registry.
- [UnregisterExtensionInfo](#) (min: 0, max: unbounded): Manages the removal of extension-related information from the system registry.
- [UnregisterFonts](#) (min: 0, max: unbounded): Removes registration information about installed fonts from the system.
- [UnregisterMIMEInfo](#) (min: 0, max: unbounded): Unregisters MIME-related registry information from the system.
- [UnregisterProgIdInfo](#) (min: 0, max: unbounded): Manages the unregistration of OLE ProgId information with the system.

- [UnregisterTypeLibraries](#) (min: 0, max: unbounded): Unregisters type libraries from the system.
- [ValidateProductID](#) (min: 0, max: unbounded): Sets the ProductID property to the full product identifier.
- [WriteEnvironmentStrings](#) (min: 0, max: unbounded): Modifies the values of environment variables.
- [WriteIniValues](#) (min: 0, max: unbounded): Writes the .ini file information that the application needs written to its .ini files.
- [WriteRegistryValues](#) (min: 0, max: unbounded): Sets up an application's registry information.

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

InstallFiles Element

Description

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

Windows Installer references

[InstallFiles Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

InstallFinalize Element

Description

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

Windows Installer references

[InstallFinalize Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

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

InstallInitialize Element

Description

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

Windows Installer references

[InstallInitialize Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

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

InstallODBC Element

Description

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

Windows Installer references

[InstallODBC Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

InstallServices Element

Description

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

Windows Installer references

[InstallServices Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

InstallUISequence Element

Description

None

Windows Installer references

[InstallUISequence Table](#)

Parents

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

Inner Text

None

Children

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

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

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

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Instance Element

Description

Defines an instance transform for your product.

Windows Installer references

None

Parents

[InstanceTransforms](#)

Inner Text

None

Children

None

Attributes

Name	Type	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](#)

Version 3.0.5419.0

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

Attributes

Name	Type	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](#)

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

IsolateComponents Element

Description

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

Windows Installer references

[IsolateComponents Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

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

Version 3.0.5419.0

LaunchConditions Element

Description

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

Windows Installer references

[LaunchConditions Action](#)

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#), [Condition](#)

Version 3.0.5419.0

Listbox Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

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

See Also

[Wix Schema](#)

Media Element

Description

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

Windows Installer references

[Media Table](#)

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [DigitalSignature](#) (min: 0, max: unbounded)
 - [PatchBaseline](#) (min: 0, max: unbounded)
 - [SymbolPath](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Integer	Disk identifier for Media table. This number must be equal to or greater than 1.	Yes
Cabinet	String	The name of the cabinet if some or all of the files stored on the media are 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	<p>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).</p>
Source	String	<p>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</p>

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.

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

See Also

[Wix Schema](#)

Version 3.0.5419.0

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)

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

Attributes

Name	Type	Description	Required
Id	String	The unique identifier for the Merge element in the source code. Referenced by the MergeRef/@Id.	Yes
DiskId	String	The value of this attribute should correspond to the Id attribute of a Media element authored elsewhere. By	

creating this connection between the merge module and Media element, you set the packaging options to the values specified in the Media element (values such as compression level, cab embedding, etc...).

FileCompression	YesNoType	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](#)

Version 3.0.5419.0

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

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

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

See Also

[Wix Schema](#)

MIME Element

Description

MIME content-type for an Extension

Windows Installer references

[MIME Table](#)

Parents

[Extension](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
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](#)

Version 3.0.5419.0

Module Element

Description

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

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

- [EnsureTable](#) (min: 0, max: unbounded)
- [Exclusion](#) (min: 0, max: unbounded)
- [Icon](#) (min: 0, max: unbounded)
- [IgnoreModularization](#) (min: 0, max: unbounded)
- [IgnoreTable](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SetDirectory](#) (min: 0, max: unbounded)
- [SetProperty](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [Substitution](#) (min: 0, max: unbounded)
- [UI](#) (min: 0, max: unbounded)
- [UIRef](#) (min: 0, max: unbounded)
- [WixVariable](#) (min: 0, max: unbounded)
- Sequence (min: 1, max: 1)
 1. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- Any Element namespace='###other'
processContents='Lax'
- [CloseApplication](#)
- [ComPlusApplication](#)
- [ComPlusApplicationRole](#)
- [ComPlusPartition](#)
- [ComPlusPartitionRole](#)
- [Group](#)
- [HelpCollectionRef](#)
- [HelpFilter](#)
- [SqlDatabase](#)
- [User](#)
- [WebApplication](#)

- [WebAppPool](#)
- [WebDirProperties](#)
- [WebLog](#)
- [WebSite](#)

Attributes

Name	Type	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 integer like 1252, or by web name like Windows-1252. See [Code Pages](#) 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	Type	Description	Required
Sequence	Integer	A value used to indicate the position of this action in a sequence.	
Suppress	YesNoType	If yes, this action will not occur.	

See Also

[Wix Schema](#)

MsiPublishAssemblies Element

Description

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

Windows Installer references

[MsiPublishAssemblies Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

MsiUnpublishAssemblies Element

Description

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

Windows Installer references

[MsiUnpublishAssemblies Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

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](#)

Version 3.0.5419.0

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

Attributes

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

Data source is registered per machine.

user

Data source is registered per user.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

ODBCTranslator Element

Description

ODBCTranslator for a Component

Windows Installer references

[ODBCTranslator Table](#)

Parents

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

Inner Text

None

Children

None

Attributes

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

See Also

[Wix Schema](#)

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	Type	Description	Required
SkipAssignment	YesNoType	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

Attributes

Name	Type	Description	Required
AdminImage	YesNoType	Set to 'yes' if the source is an admin image.	
Comments	String	Optional comments for browsing.	
Compressed	YesNoType	Set to 'yes' to have compressed files in the source. 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 must be set to 200 or greater.
------------------	---------	--

InstallPrivileges	Enumeration	<p>Use this attribute to specify the privileges required to install the package on Windows Vista and above. This attribute's value must be one of the following:</p> <p><i>limited</i></p> <p>Set this value to declare that the package does not require elevated privileges to install.</p> <p><i>elevated</i></p> <p>Set this value to declare that the package requires elevated privileges to install. This is the default value.</p>
InstallScope	Enumeration	<p>Use this attribute to specify the installation scope of this package: per-machine or per-user. This attribute's value must be one of the following:</p> <p><i>perMachine</i></p>

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. This value requires that the `InstallerVersion` property be set to 200 or greater.

x64

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 read-only. A database editing tool should not modify a read-only enforced database and should issue a warning at attempts to modify a read-only recommended database.
ShortNames	YesNoType	Set to 'yes' to have short filenames in the source.
SummaryCodepage	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 integer like 1252, or by web name like Windows-1252. See [Code Pages](#) for more information.

See Also

[Wix Schema](#)

Version 3.0.5419.0

PackageCertificates Element

Description

Digital signatures that identify installation packages in a multi-product transaction.

Windows Installer references

[MsiPackageCertificate Table](#)

Parents

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

Inner Text

None

Children

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

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

Attributes

None

See Also

[Wix Schema](#)

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)
 - [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)
 - [PatchProperty](#) (min: 0, max: unbounded)
 - [TargetProductCodes](#) (min: 0, max: unbounded)
 - [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	
AllowRemoval	YesNoType	U

ApiPatchingSymbolNoFailuresFlag	YesNoType	F V e r s i o n i n f o r m a t i o n
ApiPatchingSymbolNoImageIhpFlag	YesNoType	F V e r s i o n i n f o r m a t i o n
ApiPatchingSymbolUndecoratedTooFlag	YesNoType	F V e r s i o n i n f o r m a t i o n
Classification	PatchClassificationType	C l a s s i f i c a t i o n
ClientPatchId	String	A r b i t r a r y s t r i n g

Codepage	String	S f i i
Comments	String	T p v r r M r r i
Description	String	C c k
DisplayName	String	E t
Id	AutogenGuid	A p s p l /
Manufacturer	String	F f c
MinorUpdateTargetRTM	YesNoType	F f i

r
l
r
t
r
n
r
c
s
i
i
t
r
r
t
v
r
t
r
l
r
r
e
k
v
l
A
r
i
s
t
A
F
f
c
l
e
t

MoreInfoURL	String
OptimizedInstallMode	YesNoType

OptimizePatchSizeForLargeFiles	YesNoType	
TargetProductName	String	

Remarks

You can specify any valid Windows code by by integer like 1252, or by web name like Windows-1252. See [Code Pages](#) 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.

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	Type	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)

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

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)
 - [PatchProperty](#) (min: 0, max: unbounded)
 - [PatchSequence](#) (min: 0, max: unbounded)
 - [ReplacePatch](#) (min: 0, max: unbounded)
 - [TargetProductCode](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
Id	Guid	PatchCreation identifier primary key for identifying
AllowMajorVersionMismatches	YesNoType	Use this to set whether versions between the u target images match. See AllowProductVersionM

		for more information.
AllowProductCodeMismatches	YesNoType	Use this to set whether code between the upgrade images match. See AllowProductCodeMismatch for more information.
CleanWorkingFolder	YesNoType	Use this to set whether should clean the temp folder after finished. See DontRemoveTempFolder for more information.
Codepage	String	The code page integer name for the resulting File Remarks for more information.
OutputPath	String	The full path, including patch package file that generated. See PatchOptions for more information.
SourceList	String	Used to locate the .msi patch if the cached copy is not available. See PatchSourceList for more information.
SymbolFlags	Int	An 8-digit hex integer combination of patch symbols flags to use when creating a patch. See ApiPatching for more information.
WholeFilesOnly	YesNoType	Use this to set whether should be included in the patch. See IncludeWholeFilesOnly for more information.

Remarks

You can specify any valid Windows code by integer like 1252, or by web name like Windows-1252. See [Code Pages](#) for more information.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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)
 - [IconRef](#) (min: 0, max: unbounded)
 - [PropertyRef](#) (min: 0, max: unbounded)
 - [UIRef](#) (min: 0, max: unbounded)
 - [Any Element namespace='##other' processContents='Lax'](#)

Attributes

Name	Type	Description	Required
Id	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.

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

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

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

See Also

[Wix Schema](#)

PatchInformation Element

Description

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

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
AdminImage	YesNoType	Source is an admin image	
Comments	String	Optional comments for browsing	
Compressed	YesNoType	Compressed files on source	
Description	String	Product full name or description	
Keywords	String	Optional keywords for browsing	

Languages	String	List of language IDs supported in package
Manufacturer	String	Vendor releasing the package
Platforms	String	List of platforms supported in package
ReadOnly	YesNoDefaultType	The value of this attribute conveys whether the package should be opened as read-only. A database editing tool should not modify a read-only enforced database and should issue a warning at attempts to modify a read-only recommended database.
ShortNames	YesNoType	Short filenames on source
SummaryCodepage	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 integer like 1252, or by web name like Windows-1252. See [Code Pages](#) for more information.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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)

- Choice of elements (min: 0, max: unbounded)
 - [CustomProperty](#) (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.

Attributes

Name	Type	Description	Re
AllowRemoval	YesNoType	Whether this is an uninstalleable patch.	Yes
Classification	PatchClassificationType	Category of update.	Yes
CreationTimeUTC	String	Creation time of the .msp file in the form mm-dd-	

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

		version of the product, or up to the most recent major upgrade patch. This property is available beginning with Windows Installer 3.1.	
MoreInfoURL	String	A URL that provides information specific to this patch. In Add/Remove Programs from XP SP2 on.	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	Yes

application or
target
product suite.

See Also

[Wix Schema](#)

Version 3.0.5419.0

PatchProperty Element

Description

A property for this patch database.

Windows Installer references

[MsiPatchMetadata Table](#)

Parents

[Patch](#), [PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	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

Attributes

Name	Type	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 be 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 be specified if the ProductCode attribute is specified.

See Also
[Wix Schema](#)

Version 3.0.5419.0

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

Attributes

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

		Only valid under a 'CreateFolder' parent.
Domain	String	
EnumerateSubkeys	YesNoType	
Execute	YesNoType	
GenericAll	YesNoType	
GenericExecute	YesNoType	
GenericRead	YesNoType	specifying this will fail to grant access
GenericWrite	YesNoType	
Notify	YesNoType	
Read	YesNoType	
ReadAttributes	YesNoType	
ReadExtendedAttributes	YesNoType	
ReadPermission	YesNoType	
Synchronize	YesNoType	
TakeOwnership	YesNoType	
Traverse	YesNoType	For a directory, the right to traverse the directory. By default, users are assigned the BYPASS_TRAVERSE_CHECK privilege, which ignores the FILE_TRAVERSE access right. Only valid under a 'CreateFolder' parent.
User	String	
Write	YesNoType	
WriteAttributes	YesNoType	
WriteExtendedAttributes	YesNoType	

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

[CreateFolder](#), [File](#), [Registry](#), [RegistryKey](#), [RegistryValue](#), [ServiceInstall](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Condition](#) (min: 0, max: 1): Optional condition that controls whether the permissions are applied.

Attributes

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

See Also

[Wix Schema](#)

Product Element

Description

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

Windows Installer references

None

Parents

[Wix](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [Package](#) (min: 1, max: 1)
2. Choice of elements (min: 0, max: unbounded)
 - [AppId](#) (min: 0, max: unbounded)
 - [Binary](#) (min: 0, max: unbounded)
 - [ComplianceCheck](#) (min: 0, max: unbounded)
 - [Component](#) (min: 0, max: unbounded)
 - [ComponentGroup](#) (min: 0, max: unbounded)
 - [Condition](#) (min: 0, max: unbounded)
 - [CustomAction](#) (min: 0, max: unbounded)
 - [CustomActionRef](#) (min: 0, max: unbounded)
 - [CustomTable](#) (min: 0, max: unbounded)
 - [Directory](#) (min: 0, max: unbounded)
 - [DirectoryRef](#) (min: 0, max: unbounded)
 - [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)
- [Icon](#) (min: 0, max: unbounded)
- [InstanceTransforms](#) (min: 0, max: unbounded)
- [Media](#) (min: 0, max: unbounded)
- [PackageCertificates](#) (min: 0, max: unbounded)
- [PatchCertificates](#) (min: 0, max: unbounded)
- [Property](#) (min: 0, max: unbounded)
- [PropertyRef](#) (min: 0, max: unbounded)
- [SetDirectory](#) (min: 0, max: unbounded)
- [SetProperty](#) (min: 0, max: unbounded)
- [SFPCatalog](#) (min: 0, max: unbounded)
- [SymbolPath](#) (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. [InstallExecuteSequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)
 3. [AdminExecuteSequence](#) (min: 0, max: 1)
 4. [AdminUISequence](#) (min: 0, max: 1)
 5. [AdvertiseExecuteSequence](#) (min: 0, max: 1)
- Any Element namespace='###other'
processContents='Lax'
- [CloseApplication](#)
- [ComPlusApplication](#)
- [ComPlusApplicationRole](#)
- [ComPlusPartition](#)
- [ComPlusPartitionRole](#)
- [Group](#)
- [HelpCollectionRef](#)

- [HelpFilter](#)
- [SqlDatabase](#)
- [User](#)
- [WebApplication](#)
- [WebAppPool](#)
- [WebDirProperties](#)
- [WebLog](#)
- [WebSite](#)

Attributes

Name	Type	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'			

Remarks

You can specify any valid Windows code by integer like 1252, or by web name like Windows-1252. See [Code Pages](#) for more information.

See Also

[Wix Schema](#)

Version 3.0.5419.0

ProgId Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [ProgId](#) (min: 0, max: unbounded): The version-independent ProgId must be the first child element of actual ProgId. Nesting other ProgId elements within the Version-independent ProgId will create COM+ aliases, see <http://support.microsoft.com/kb/305745> for more information.
2. [Extension](#) (min: 0, max: unbounded): Extensions that refer to this ProgId

Attributes

Name	Type	Description	Required
Id	String		Yes
Advertise	YesNoType		
Description	String		
Icon	String	For an advertised ProgId, the Id of an Icon element. For a non-advertised ProgId, this is the Id of a file containing an icon resource.	

IconIndex	Integer	
NoOpen	String	Specifies that the associated ProgId should not be opened by users. The value is presented as a warning to users. An empty string is also valid for this attribute.

See Also

[Wix Schema](#)

Version 3.0.5419.0

ProgressText Element

Description

None

Windows Installer references

[ActionText Table](#)

Parents

[UI](#)

Inner Text (xs:string)

Element value is progress message text for action

Children

None

Attributes

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

See Also

[Wix Schema](#)

Property Element

Description

Property value for a Product or Module.

Windows Installer references

[Property Table](#)

Parents

[Control](#), [Fragment](#), [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)
 1. [ComplianceDrive](#) (min: 0, max: 1): Starts searches from the CCP_DRIVE.
 2. [ComponentSearch](#) (min: 0, max: unbounded)
 3. [RegistrySearch](#) (min: 0, max: unbounded)
 4. [RegistrySearchRef](#) (min: 0, max: unbounded)
 5. [IniFileSearch](#) (min: 0, max: unbounded)
 6. [DirectorySearch](#) (min: 0, max: unbounded)
 7. [DirectorySearchRef](#) (min: 0, max: unbounded)
- [Any Element namespace='###other' processContents='Lax'](#)

Attributes

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

		AdminProperties Property for more information.
ComplianceCheck	YesNoType	Adds a row to the CCPSearch table. This attribute is only valid when this Property contains a search element.
Hidden	YesNoType	Denotes that the Property is not logged during installation. See the MsiHiddenProperties Property for more information.
Secure	YesNoType	Denotes that the Property can be passed to the server side when doing a managed installation with elevated privileges. See the SecureCustomProperties Property for more information.
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'

How Tos and Examples

- [How To: Check the version number of a file during installation](#)

See Also

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

Version 3.0.5419.0

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	Type	Description	Required
Id	String	Identifier of Property to reference.	Yes
Any attribute namespace='###other' processContents='lax'			

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

See Also

[Wix Schema](#)

Publish Element

Description

None

Windows Installer references

[ControlEvents Table](#)

Parents

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

Inner Text (xs:string)

The element value is the optional Condition expression.

Children

None

Attributes

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

See Also

[Wix Schema](#)

PublishFeatures Element

Description

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

Windows Installer references

[PublishFeatures Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

PublishProduct Element

Description

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

Windows Installer references

[PublishProduct Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RadioButton Element

Description

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

Windows Installer references

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

Parents

[RadioButtonGroup](#)

Inner Text

None

Children

None

Attributes

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

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

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

See Also

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

RadioButtonGroup Element

Description

Set of radio buttons tied to the specified Property

Windows Installer references

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

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

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

See Also

[Wix Schema](#)

RegisterComPlus Element

Description

Registers COM+ applications. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterComPlus Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterExtensionInfo Element

Description

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

Windows Installer references

[RegisterExtensionInfo Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterFonts Element

Description

Registers installed fonts with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterFonts Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterMIMEInfo Element

Description

Registers MIME-related registry information with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterMIMEInfo Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterProduct Element

Description

Registers the product information with the installer. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterProduct Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterProgIdInfo Element

Description

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

Windows Installer references

[RegisterProgIdInfo Action](#)

Parents

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

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterTypeLibraries Element

Description

Registers type libraries with the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterTypeLibraries Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RegisterUser Element

Description

Registers the user information with the installer to identify the user of a product. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RegisterUser Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Registry Element

Description

This element has been deprecated; please use the [RegistryValue](#) 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)
- [PermissionEx](#) (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'](#)
- [PermissionEx](#)

Attributes

Name	Type	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: <i>append</i> Appends the specified value(s) to a multiString registry key. <i>createKey</i> Creates the key, if absent, when the parent component is

installed.

createKeyAndRemoveKeyOnUninstall

Creates the key, if absent, when the parent component is installed then remove the key with all its values and subkeys when the parent component is uninstalled.

prepend

Prepends the specified value(s) to a multiString registry key.

remove

Removes a registry name when the parent component is installed.

removeKeyOnInstall

Removes a key with all its values and subkeys when the parent component is installed.

removeKeyOnUninstall

Removes a key with all its values and subkeys when the parent component is uninstalled.

write

Writes a registry value.

Id	String	Primary key used to identify this particular entry. If this attribute is not specified, an identifier will be generated by hashing the parent Component identifier, Root, Key, and Name.
Key	String	The localizable key for the registry value.
KeyPath	YesNoType	Set this attribute to 'yes' to make this registry key the KeyPath of the parent

component. Only one resource (registry, file, etc) can be the KeyPath of a component.

Name	String	The localizable registry value name. If this attribute is not provided the default value for the registry key will be set instead. The Windows Installer allows several special values to be set for this attribute. You should not use them in WiX. Instead use appropriate values in the Action attribute to get the desired behavior.
Root	RegistryRootType	The predefined root key for the registry value.
Type	Enumeration	<p>Set this attribute to the type of the desired registry key. This attribute must be specified whenever the Value attribute or a child RegistryValue element is specified. This attribute should only be set when the value of the Action attribute does not include the word 'remove'. This attribute's value must be one of the following:</p> <p><i>string</i> The value is interpreted and stored as a string (REG_SZ).</p> <p><i>integer</i> The value is interpreted and stored as an integer (REG_DWORD).</p> <p><i>binary</i> The value is interpreted and stored as a hexadecimal value (REG_BINARY).</p> <p><i>expandable</i> The value is interpreted and</p>

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
- [PermissionEx](#) (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'](#)
- [PermissionEx](#)

Attributes

Name	Type	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: <i>create</i> Creates the key, if 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.

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.
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](#)

Version 3.0.5419.0

RegistrySearch Element

Description

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

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	String	Signature to be used for the file, directory or registry key being searched for.	Yes
Key	String	Key for the registry value.	Yes
Name	String	Registry value name.	
Root	Enumeration	Root key for the registry value. This attribute's value must be one of the following: <i>HKCR</i> HKEY_CLASSES_ROOT <i>HKCU</i> HKEY_CURRENT_USER	Yes

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:	Yes
------	-------------	--	-----

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 followed by '+' or '-'.

REG_BINARY

Starts with '#x' and the installer converts and saves each hexadecimal digit

(nibble) as an ASCII character prefixed by '#x'.

REG_EXPAND_SZ

Starts with '#%'.

REG_MULTI_SZ

Starts with '[~]' and ends with '[~]'.

REG_SZ

No prefix, but if the first character of the registry value is '#', the installer escapes the character by prefixing it with another '#'.

Win64	YesNoType	Instructs the search to look in the 64-bit registry when the value is 'yes'. Default is 'no' and search looks in the 32-bit registry.
-------	---------------------------	---

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](#)

Version 3.0.5419.0

RegistrySearchRef Element

Description

References an existing RegistrySearch element.

Windows Installer references

None

Parents

[Property](#)

Inner Text

None

Children

None

Attributes

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

See Also

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

Version 3.0.5419.0

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)
- [PermissionEx](#) (min: 0, max: unbounded): Can also configure the ACLs for this registry value.
- [Any Element namespace='###other' processContents='Lax'](#)
- [PermissionEx](#)

Attributes

Name	Type	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 parent.
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: <i>string</i> 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](#)

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

RemoveEnvironmentStrings Element

Description

Modifies the values of environment variables. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RemoveEnvironmentStrings Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RemoveExistingProducts Element

Description

Goes through the product codes listed in the ActionProperty column of the Upgrade table and removes the products in sequence. Special actions don't have a built-in sequence number and thus must appear relative to another action. The suggested way to do this is by using the Before or After attribute. InstallExecute and InstallExecuteAgain can optionally appear anywhere between InstallInitialize and InstallFinalize.

Windows Installer references

[RemoveExistingProducts Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

RemoveFile Element

Description

Remove a file(s) if the parent component is selected for installation or removal. Multiple files can be removed by specifying a wildcard for the value of the Name attribute. By default, the source directory of the file is the directory of the parent component. This can be overridden by specifying the Directory attribute with a value corresponding to the Id of the source directory, or by specifying the Property attribute with a value corresponding to a property that will have a value that resolves to the full path to the source directory.

Windows Installer references

[RemoveFile Table](#)

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
Id	String	Primary key used to identify this particular entry.
Directory	String	Overrides the directory of the parent component with a specific Directory. This Directory must exist in the installer database at creation time. This attribute cannot be specified in conjunction with the

		Property attribute.
LongName	WildcardLongFileNameType	This attribute has been deprecated; please use the ShortName attribute instead.
Name	WildcardLongFileNameType	This value should be set to the localizable name of the file(s) to be removed. All the files that match the wildcard will be removed from the specified directory. The value is a filename that may also contain the wildcard 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 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. 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 name. However, if you want to manually specify the short file name, then the ShortName attribute may be specified.
On	Enumeration	This value determines the

time at which the file(s) n
be removed. This attribut
value must be one of the
following:

install

Removes the file on
when the parent
component is being
installed

(msiInstallStateLoca
msiInstallStateSourc

uninstall

Removes the file on
when the parent
component is being
removed

(msiInstallStateAbse

both

Removes the file wh
the parent compone
is being installed or
removed.

Property String

Overrides the directory o
the parent component wi
the value of the specified
property. The property
should have a value that
resolves to the full path c
the source directory. The
property does not have to
exist in the installer
database at creation time
could be created at
installation time by a cus
action, on the command
line, etc. This attribute
cannot be specified in

	conjunction with the Directory attribute.
ShortName WildcardShortFileNameType	The short file name of the file in 8.3 format. This attribute should only be used if you want to manually specify the short file name.

See Also

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

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

RemoveFolder Element

Description

Remove an empty folder if the parent component is selected for installation or removal. By default, the folder is the directory of the parent component. This can be overridden by specifying the Directory attribute with a value corresponding to the Id of the directory, or by specifying the Property attribute with a value corresponding to a property that will have a value that resolves to the full path of the folder.

Windows Installer references

[RemoveFile Table](#)

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary key used to identify this particular entry.	Yes
Directory	String	Overrides the directory of the parent component with a specific Directory. This Directory must exist in the installer database at creation time. This attribute cannot be specified in conjunction with the Property attribute.	
On	Enumeration	This value determines the time at which the folder may	Yes

be removed. This attribute's value must be one of the following:

install

Removes the folder only when the parent component is being installed (msiInstallStateLocal or msiInstallStateSource).

uninstall

Removes the folder only when the parent component is being removed (msiInstallStateAbsent).

both

Removes the folder when the parent component is being installed or removed.

Property String	Overrides the directory of the parent component with the value of the specified property. The property should have a value that resolves to the full path of the source directory. The property does not have to exist in the installer database at creation time; it could be created at installation time by a custom action, on the command line, etc. This attribute cannot be specified in conjunction with the Directory attribute.
-----------------	---

See Also

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

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

RemoveIniValues Element

Description

Removes .ini file information specified for removal in the RemoveIniFile table if the component is set to be installed locally or run from source. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RemoveIniValues Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

RemoveODBC Element

Description

Removes the data sources, translators, and drivers listed for removal during the installation. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RemoveODBC Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

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

Attributes

Name	Type	Description	Required
Action	Enumeration	<p>This is the action that will be taken for this registry value. This attribute's value must be one of the following:</p> <p><i>removeOnInstall</i> Removes a key with all its values and subkeys when the parent component is installed.</p> <p><i>removeOnUninstall</i> Removes a key with all its values and subkeys when the parent component is uninstalled.</p>	

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.
Root	RegistryRootType	The predefined root key for the registry value.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

Attributes

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

See Also

[Wix Schema](#)

RemoveShortcuts Element

Description

Manages the removal of an advertised shortcut whose feature is selected for uninstallation or a nonadvertised shortcut whose component is selected for uninstallation. The condition for this action may be specified in the element's inner text.

Windows Installer references

[RemoveShortcuts Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

ReplacePatch Element

Description

A patch that is deprecated by this patch.

Windows Installer references

None

Parents

[PatchCreation](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	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 [privilege constant](#) or a Formatted property that resolves to a privilege constant.

Windows Installer references

[MsiServiceConfig Table](#)

Parents

[ServiceConfig](#)

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

Attributes

Name	Type	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](#)

Version 3.0.5419.0

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

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

RMCCPSearch Element

Description

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

Windows Installer references

[RMCCPSearch Action](#)

Parents

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

Inner Text (xs:string)

Text node specifies the condition of the action.

Children

None

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#), [CCPSearch](#), [ComplianceCheck](#)

Version 3.0.5419.0

Row Element

Description

Row data for a Custom Table

Windows Installer references

None

Parents

[CustomTable](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

None

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

Attributes

Name	Type	Description	Required
After	String	The name of an action that this action should come after.	
Before	String	The name of an action that this action should come before.	
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](#)

Version 3.0.5419.0

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

See Also

[Wix Schema](#)

SelfUnregModules Element

Description

Unregisters all modules listed in the SelfReg table that are scheduled to be uninstalled. The condition for this action may be specified in the element's inner text.

Windows Installer references

[SelfUnregModules Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

ServiceArgument Element

Description

Argument used in ServiceControl parent

Windows Installer references

[ServiceControl Table](#)

Parents

[ServiceControl](#)

See Also

[Wix Schema](#)

Version 3.0.5419.0

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.

Attributes

Name	Type	Description	R
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.

Attributes

Name	Type	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](#)

Version 3.0.5419.0

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. [ServiceArgument](#) (min: 0, max: unbounded): Ordered list of arguments used when modifying services.

Attributes

Name	Type	Description	Required
Id	String		Yes
Name	String	Name of the service.	Yes
Remove	Enumeration	Specifies whether the service should be removed on install, uninstall or both. This attribute's value must be one of the following: <i>install</i> The service will be deleted by the	

DeleteServices action during install.

uninstall

The service will be deleted by the DeleteServices action during uninstall.

both

The service will be deleted by the DeleteServices action during install and uninstall.

Start	Enumeration	Specifies whether the service should be started on install, uninstall or both. This attribute's value must be one of the following: <i>install</i> The service will be started by the StartServices action during install. <i>uninstall</i> The service will be started by the StartServices action during uninstall. <i>both</i> The service will be started by the StartServices action during install and uninstall.
Stop	Enumeration	Specifies whether the service should be stopped on install, uninstall or both. This attribute's value must be one

of the following:

install

The service will be stopped by the StopServices action during install.

uninstall

The service will be stopped by the StopServices action during uninstall.

both

The service will be stopped by the StopServices action during install and uninstall.

Wait	YesNoType	Specifies whether or not to wait for the service to complete before continuing.
------	---------------------------	---

See Also

[Wix Schema](#)

ServiceDependency Element

Description

Service or group of services that must start before the parent service.

Windows Installer references

[ServiceInstall Table](#)

Parents

[ServiceInstall](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The value of this attribute should be one of the following: <ol style="list-style-type: none">1. The name (not the display name) of a previously installed service.2. A foreign key referring to another ServiceInstall/@Id.3. A group of services (in which case the Group attribute should be set to 'yes').	Yes
Group	YesNoType	Set to 'yes' to indicate that the value in the Id attribute is the name of a group of services.	

See Also

[Wix Schema](#)

Version 3.0.5419.0

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)

- [PermissionEx](#) (min: 0, max: unbounded): Configures the ACLs for this service.
- [ServiceConfig](#) (min: 0, max: unbounded)
- [ServiceConfigFailureActions](#) (min: 0, max: unbounded)
- [ServiceDependency](#) (min: 0, max: unbounded): ordered list of dependencies when installing services
- [Any Element namespace='###other' processContents='Lax'](#)
- [ServiceConfig](#)

Attributes

Name	Type	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: <i>ignore</i> Logs the error and continues with the startup operation. <i>normal</i> Logs the error, displays a message box and continues the startup operation. <i>critical</i> Logs the error if it is possible and the system is	Yes

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 support boot or system. This attribute's value must	Yes

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 IoInitSystem 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: <i>ownProcess</i> A Win32 service that runs its own process. <i>shareProcess</i> A Win32 service that shares a process. <i>kernelDriver</i> A kernel driver service. This value is not currently supported by the Windows Installer. <i>systemDriver</i> 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](#)

Version 3.0.5419.0

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

Attributes

Name	Type	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 Enumeration		Controls which sequences the Directory assignment is sequenced in. The default is both. This attribute's value must be one of the following: <i>both</i> Schedules the assignment in the InstallUISequence and	

the
InstallExecuteSequence.

execute

Schedules the
assignment only in the
the
InstallExecuteSequence.

ui

Schedules the
assignment only in the
the InstallUISequence.

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'

See Also

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

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	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](#)

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

Attributes

Name	Type	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 Enumeration		Controls which sequences the Property assignment is sequenced in. The default is both. This attribute's value must be one of the following: <i>both</i> Schedules the assignment in the InstallUISequence and the InstallExecuteSequence. <i>execute</i> Schedules the assignment only in the the InstallExecuteSequence. <i>ui</i> Schedules the assignment only in the the InstallUISequence.
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'

See Also

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

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	Type	Description	Required
Dependency	String	Used to define dependency outside of the package.	
Name	String	Filename for catalog file when installed.	
SourceFile	String	Path to catalog file in binary.	

See Also

[Wix Schema](#)

SFPFile Element

Description

Provides a many-to-many mapping from the SFPCatalog table to the File table

Windows Installer references

[FileSFPCatalog Table](#)

Parents

[SFPCatalog](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key to File Table.	Yes

See Also

[Wix Schema](#)

Shortcut Element

Description

Shortcut, default target is parent File, CreateFolder, or Component's Directory

Windows Installer references

[Shortcut Table](#)

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description
Id	String	Unique identifier for the shortcut. This value was the primary key for
Advertise	YesNoType	Specifies if the shortcut be advertised or not. For advertised shortcuts a point at a particular application identified by a Product and should not be shared between applications. Advertised shortcuts are for the most recently installed application, and are removed when that application is removed. The default

		'no'.
Arguments	String	The command-line argument for the shortcut. Note the resolution of properties in the Arguments field is limited. A property formatted as [property] in this field can only be resolved if the property already has an intended value when the component owning the shortcut is installed. For example, the argument "[#MyDoc.doc]" will resolve to the correct file MyDoc.doc only if the same process must be running as the file MyDoc.doc and the component that owns the shortcut.
Description	String	The localizable description of the shortcut.
DescriptionResourceDll	String	The Formatted string ID of the full path to the language-neutral file containing the Manifest. Generally using the [filekey] form. If the attribute is specified, the DescriptionResourceDll must also be provided. This attribute is only used on Windows Vista and above. If this attribute is not specified and the install is running on Windows Vista and above, the value of the Name attribute is used. If this attribute is provided and the install is running on Windows Vista and above, the value in the attribute is ignored.

DescriptionResourceId Integer

The description name the shortcut. This must be a non-negative number. When this attribute is specified, the DescriptionResourceId attribute must also be populated.

This attribute is only used in Windows Vista and above. When this attribute is not specified and the install is running on Windows Vista and above, the value of the Name attribute is used. When this attribute is populated and the install is running on Windows Vista and above, the value of the Name attribute is ignored.

Directory	String	Identifier reference to the Component element where the shortcut was created. When nested in a Component element, the attribute's value will default to the parent directory. This attribute is required.
-----------	--------	---

DisplayResourceDll	String	The Formatted string representing the full path to the language-neutral file containing the Manifest. Generally using the [#filekey] form. When this attribute is specified, the DescriptionResourceId attribute must also be provided.
--------------------	--------	---

This attribute is only used in Windows Vista and above. When this attribute is not populated and the install is running

		<p>Vista and above, the value of the Name attribute is ignored. This attribute is populated by the installer when the installation is running on Windows Vista and above, the value of the Name attribute is ignored.</p>
DisplayResourceId	Integer	<p>The display name index for the shortcut. This must be a negative number. When this attribute is specified, the DisplayResourceDll attribute must also be provided.</p> <p>This attribute is only used on Windows Vista and above. When this attribute is not specified, the Name attribute is used. This attribute is not populated by the installer when the installation is running on Windows Vista and above, the value of the Name attribute is ignored.</p>
Hotkey	Integer	<p>The hotkey for the shortcut. The low-order byte contains the virtual-key code for the key, and the high-order byte contains the modifier flags. This must be a non-negative number. Most installation packages generally recommend not using this option, because it can add duplicate hotkeys to the user's desktop. In addition, the practice of assigning hotkeys to shortcuts can be problematic for users using hotkeys for accessibility.</p>

Icon	String	Identifier reference to element. The Icon id should have the same extension as the file it points at. For example shortcut to an executable ("my.exe") should refer to Icon with identifier like "MyIcon.exe"
IconIndex	Integer	Identifier reference to element.
LongName	LongFileNameType	This attribute has been deprecated; please use Name attribute instead.
Name	LongFileNameType	In prior versions of the toolset, this attribute specified the short name. This attribute 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 attribute is not specified, the ShortName attribute may be omitted to allow WiX to attempt to generate a unique short name. However, if this name conflicts with another shortcut, you may wish to manually specify a short name, then the ShortName attribute may not be specified.
ShortName	ShortFileNameType	The short name of the element in 8.3 format. This attribute should only be set if there is a conflict between generated

		short names or the us to manually specify th name.
Show	Enumeration	<p>This attribute's value r one of the following:</p> <p><i>normal</i> The shortcut targ displayed using th SW_SHOWNORI attribute.</p> <p><i>minimized</i> The shortcut targ displayed using th SW_SHOWMINN attribute.</p> <p><i>maximized</i> The shortcut targ displayed using th SW_SHOWMAXI attribute.</p>
Target	String	<p>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. T attribute's value is the a non-advertised shor attribute is not valid fo advertised shortcuts. I specify this value, its \ should be a property i enclosed by square br]), that is expanded in or a folder pointed to l shortcut.</p>

WorkingDirectory

String

Directory identifier (or identifier that resolves directory) that resolve path of the working dir the shortcut.

How Tos and Examples

- [How To: Create a shortcut on the Start Menu](#)

See Also

[Wix Schema](#)

Version 3.0.5419.0

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	Type	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	Type	Description	Required
After	String		
Before	String		
Dialog	String		Yes
OnExit	Enumeration	mutually exclusive with Before, After, and Sequence attributes This attribute's value must be one of the following: <i>success</i> <i>cancel</i> <i>error</i> <i>suspend</i>	
Overridable	YesNoType	If "yes", the sequencing of this dialog may be overridden by sequencing	

elsewhere.

Sequence Integer

See Also

[Wix Schema](#)

Version 3.0.5419.0

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

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

StopServices Element

Description

Stops system services. The condition for this action may be specified in the element's inner text.

Windows Installer references

[StopServices Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Subscribe Element

Description

Sets attributes for events in the EventMapping table

Windows Installer references

[EventMapping Table](#)

Parents

[Control](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Attribute	String	if not present can only handle enable, disable, hide, unhide events	
Event	String	must be one of the standard control events'	

See Also

[Wix Schema](#)

Substitution Element

Description

Specifies the configurable fields of a module database and provides a template for the configuration of each field.

Windows Installer references

None

Parents

[Module](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Column	String	Specifies the target column in the row named in the Row column.	Yes
Row	String	Specifies the primary keys of the target row in the table named in the Table column. If multiple keys, separated by semicolons.	Yes
Table	String	Specifies the name of the table being modified in the module database.	Yes
Value	String	Provides a formatting template for the data being substituted into the target field specified by Table, Row, and Column.	

See Also

[Wix Schema](#)

SymbolPath Element

Description

A path to symbols.

Windows Installer references

None

Parents

[Component](#), [Directory](#), [ExternalFile](#), [File](#), [Media](#), [Product](#),
[TargetFile](#), [TargetImage](#), [UpgradeFile](#), [UpgradedImage](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Path	String	The path.	Yes

See Also

[Wix Schema](#)

TargetFile Element

Description

Information about specific files in a target image.

Windows Installer references

None

Parents

[TargetImage](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

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

Attributes

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

See Also

[Wix Schema](#)

TargetImage Element

Description

Contains information about the target images of the product.

Windows Installer references

None

Parents

[UpgradeImage](#)

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	String	Identifier for the target image.	Yes
IgnoreMissingFiles	YesNoType	Files missing from the target image are ignored by the installer.	
Order	Int	Relative order of the target image.	Yes
SourceFile	String	Full path to the location of the msi file for the target image.	
src	String	This attribute has been deprecated; please use the SourceFile attribute	

		instead.
Validation	String	Product checking to avoid applying irrelevant transforms.

See Also

[Wix Schema](#)

Version 3.0.5419.0

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	Type	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)

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

Attributes

Name	Type	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	Type	Description	Required
SourceFile	String	Instructs the text to be imported from a file instead of the element value during the binding process.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	

See Also

[Wix Schema](#)

TextStyle Element

Description

None

Windows Installer references

[TextStyle Table](#)

Parents

[UI](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Blue	Integer	0 to 255	
Bold	YesNoType		
FaceName	String		Yes
Green	Integer	0 to 255	
Italic	YesNoType		
Red	Integer	0 to 255	
Size	String		Yes
Strike	YesNoType		
Underline	YesNoType		

See Also

[Wix Schema](#)

TypeLib Element

Description

Register a type library (TypeLib). Please note that in order to properly use this non-advertised, you will need use this element with Advertise='no' and also author the appropriate child Interface elements by extracting them from the type library itself.

Windows Installer references

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

Parents

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

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
Id	Guid	The GUID that identifies the type library.	Yes
Advertise	YesNoType	Value of 'yes' will create a row in the TypeLib table. Value of 'no' will create rows in the Registry table. 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)

- [BillboardAction](#) (min: 0, max: unbounded): Billboard table item with child Controls
- [Binary](#) (min: 0, max: unbounded)
- [ComboBox](#) (min: 0, max: unbounded): ComboBox table with ListItem children
- [Dialog](#) (min: 0, max: unbounded): Dialog specification, called from Sequence
- [DialogRef](#) (min: 0, max: unbounded): Reference to a Dialog specification.
- [EmbeddedUI](#) (min: 0, max: unbounded): Embedded UI definition with EmbeddedResource children.
- [Error](#) (min: 0, max: unbounded): Error text associated with install error
- [ListBox](#) (min: 0, max: unbounded): ListBox table with ListItem children
- [ListView](#) (min: 0, max: unbounded): ListView table with ListItem children
- [ProgressText](#) (min: 0, max: unbounded): ActionText entry associated with an action
- [Property](#) (min: 0, max: unbounded)

- [PropertyRef](#) (min: 0, max: unbounded)
- [Publish](#) (min: 0, max: unbounded)
- [RadioButtonGroup](#) (min: 0, max: unbounded): RadioButton table with RadioButton children
- [TextStyle](#) (min: 0, max: unbounded): TextStyle entry for use in control text
- [UIRef](#) (min: 0, max: unbounded)
- [UIText](#) (min: 0, max: unbounded): values for UIText property, not installer Property
- Sequence (min: 1, max: 1)
 1. [AdminUISequence](#) (min: 0, max: 1)
 2. [InstallUISequence](#) (min: 0, max: 1)

Attributes

Name	Type	Description	Required
Id	String		

See Also

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

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	Type	Description	Required
Id	String		Yes
Any attribute namespace='###other' processContents='lax'			

See Also

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

UIText Element

Description

Text associated with certain controls

Windows Installer references

[UIText Table](#)

Parents

[UI](#)

Inner Text (xs:string)

Element value is text, may use CDATA if needed to escape XML delimiters

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes

See Also

[Wix Schema](#)

UnpublishComponents Element

Description

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

Windows Installer references

[UnpublishComponents Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnpublishFeatures Element

Description

Removes selection-state and feature-component mapping information from the registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnpublishFeatures Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterClassInfo Element

Description

Manages the removal of COM class information from the system registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterClassInfo Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterComPlus Element

Description

Removes COM+ applications from the registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterComPlus Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterExtensionInfo Element

Description

Manages the removal of extension-related information from the system registry. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterExtensionInfo Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterFonts Element

Description

Removes registration information about installed fonts from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterFonts Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterMIMEInfo Element

Description

Unregisters MIME-related registry information from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterMIMEInfo Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterProgIdInfo Element

Description

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

Windows Installer references

[UnregisterProgIdInfo Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

UnregisterTypeLibraries Element

Description

Unregisters type libraries from the system. The condition for this action may be specified in the element's inner text.

Windows Installer references

[UnregisterTypeLibraries Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

Upgrade Element

Description

Upgrade info for a particular UpgradeCode

Windows Installer references

[Upgrade Table](#)

Parents

[Fragment](#), [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.
- [UpgradeVersion](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	Guid	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

[UpgradeImage](#)

Inner Text

None

Children

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

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

Attributes

Name	Type	Description	Required
AllowIgnoreOnError	YesNoType	Specifies whether patching this file is vital.	
File	String	Foreign key into the File table.	Yes
Ignore	YesNoType	If yes, the file is ignored during patching, and the next two attributes are ignored.	Yes
WholeFile	YesNoType	Whether the whole file should be installed, rather than creating a binary patch.	

See Also

[Wix Schema](#)

Version 3.0.5419.0

UpgradeImage Element

Description

Contains information about the upgraded images of the product.

Windows Installer references

None

Parents

[Family](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. [TargetImage](#) (min: 1, max: unbounded)
2. Choice of elements (min: 0, max: unbounded)
 - [SymbolPath](#) (min: 0, max: unbounded)
 - [UpgradeFile](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier to connect target images with upgraded image.	Yes
SourceFile	String	Full path to location of msi file for upgraded image.	
SourcePatch	String	Modified copy of the upgraded installation database that contains additional authoring specific to patching.	
src	String	This attribute has been deprecated; please use the SourceFile attribute instead.	
srcPatch	String	This attribute has been deprecated; please use the	

SourcePatch attribute instead.

See Also

[Wix Schema](#)

Version 3.0.5419.0

UpgradeVersion Element

Description

None

Windows Installer references

[Upgrade Table](#)

Parents

[Upgrade](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

Name	Type	Description	Required
ExcludeLanguages	YesNoType	Set to "yes" to detect all languages, excluding the languages listed in the Language attribute.	
IgnoreRemoveFailure	YesNoType	Set to "yes" to continue installation upon failure to remove a product or application.	
IncludeMaximum	YesNoType	Set to "yes" to make the range of versions detected include the value specified in Maximum.	
IncludeMinimum	YesNoType	Set to "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	Yes

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.

RemoveFeatures	String	The installer sets the REMOVE property to features specified in this column. The features to be removed can be determined at run time. The Formatted string entered in this field must
----------------	--------	--

evaluate to a comma-delimited list of feature names. For example: [Feature1],[Feature2],[Feature3]. No features are removed if the field contains formatted text that evaluates to an empty string. The installer sets REMOVE=ALL only if the Remove field is empty.

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

See Also

[Wix Schema](#)

Version 3.0.5419.0

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	Type	Description	Required
IgnoreAddExistingRow	YesNoType	Ignore errors when adding existing rows. The default is 'yes'.	
IgnoreAddExistingTable	YesNoType	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	YesNoType	Ignore errors when deleting missing rows. The default is	

		'yes'.
IgnoreDeleteMissingTable	YesNoType	Ignore errors when deleting missing tables. The default is 'yes'.
IgnoreUpdateMissingRow	YesNoType	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: <i>Major</i> 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: <i>Lesser</i> Installed ProductVersion < target ProductVersion. <i>LesserOrEqual</i> Installed ProductVersion <= target ProductVersion. <i>Equal</i> Installed
------------------------	-------------	--

ProductVersion
= target
ProductVersion.

GreaterOrEqual
Installed
ProductVersion
>= target
ProductVersion.

Greater
Installed
ProductVersion
> target
ProductVersion.

UpgradeCode	YesNoType	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 properties 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](#)

Version 3.0.5419.0

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

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

See Also

[Wix Schema](#)

Version 3.0.5419.0

Verb Element

Description

Verb definition for an Extension. When advertised, this element creates a row in the [Verb table](#). When not advertised, this element creates the appropriate rows in [Registry table](#).

Windows Installer references

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

Parents

[Extension](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The verb for the command.	Yes
Argument	String	Value for the command arguments. Note that the resolution of properties in the Argument field is limited. A property formatted as [Property] in this field can only be resolved if the property already has the intended value when the component owning the verb is installed. For example, for the argument "[#MyDoc.doc]" to resolve to the correct value, the same process must be installing the file	

MyDoc.doc and the component that owns the verb.

Command	String	The localized text displayed on the context menu.
Sequence	Integer	The sequence of the commands. Only verbs for which the Sequence is specified are used to prepare an ordered list for the default value of the shell key. The Verb with the lowest value in this column becomes the default verb. Used only for Advertised verbs.
Target	String	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 Schema](#)

Wix Element

Description

This is the top-level container element for every wxs file. Among the possible children, the 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)
 - [Module](#) (min: 0, max: 1)
 - [Patch](#) (min: 0, max: 1)
 - [Product](#) (min: 0, max: 1)
 2. [Fragment](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
RequiredVersion	VersionType	Required version of the WiX toolset for this input file.

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

RequiredVersion String

The version of this extension require
compile the defining source.

(<http://schemas.microsoft.com/wix/P>)

See Also

[Wix Schema](#)

Version 3.0.5419.0

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; its a WiX-only concept.

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

Name	Type	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). The default value	

		is 'no'.	
Value	String	The value of the variable. The value cannot be an empty string because that would make it possible to accidentally set a column to null.	Yes

See Also

[Wix Schema](#)

Version 3.0.5419.0

WriteEnvironmentStrings Element

Description

Modifies the values of environment variables. The condition for this action may be specified in the element's inner text.

Windows Installer references

[WriteEnvironmentStrings Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

WriteIniValues Element

Description

Writes the .ini file information that the application needs written to its .ini files. The condition for this action may be specified in the element's inner text.

Windows Installer references

[WriteIniValues Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

WriteRegistryValues Element

Description

Sets up an application's registry information. The condition for this action may be specified in the element's inner text.

Windows Installer references

[WriteRegistryValues Action](#)

Parents

[InstallExecuteSequence](#)

Inner Text (xs:string)

This element may have inner text.

Children

None

Attributes

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

See Also

[Wix Schema](#)

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Wix Schema](#)

ComponentGuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF" or "{01234567-89AB-CDEF-0123-456789ABCDEF}", but also allows "PUT-GUID-HERE" for use in examples. It's also possible to have an empty value "".

Pattern Type

Must match the regular expression: '[(]?[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}[)]?|PUT\-[GUID\-(?:\d+|-)?HERE|([!\$])\(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_]*\)|\)*|^\$'.

See Also

[Wix Schema](#)

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

[Wix Schema](#)

HexType (Simple Type)

Description

This type supports any hexadecimal number. Both upper and lower case is acceptable for letters appearing in the number. This type also includes the empty string: "".

Pattern Type

Must match the regular expression: '[0-9A-Fa-f]*'.

See Also

[Wix Schema](#)

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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Wix Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.extension". 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

[Wix Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Wix Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Wix 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: `'[^\\?|><:\^*" \+,;=\[\] \.]{1,8}(\. [^\\?|><:\^*" \+,;=\[\] \.]{0,3})?(![!$])\(\loc\.[_A-Za-z][0-9A-Za-z_]*\)'`.

See Also

[Wix Schema](#)

VersionType (Simple Type)

Description

Values of this type will look like: "x.x.x.x" where x is an integer from 0 to 65534.

Pattern Type

Must match the regular expression: `'(\d{1,5}\.){3}\d{1,5}'`.

See Also

[Wix Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Wix Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: '[^\\|><:"'+,;=\\[\.]{1,16}\\. [^\\|><:"'+,;=\\[\.]{0,6}?|(!\$)\\(loc\\. [_A-Za-z][0-9A-Za-z_\\.]*\\)'

See Also

[Wix Schema](#)

YesNoDefaultType (Simple Type)

Description

Values of this type will either be "default", "yes", or "no".

Enumeration Type

Possible values: {default, no, yes}

See Also

[Wix Schema](#)

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"?>`
`<WixLocalization`
`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

Attributes

Name	Type	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](#)

Version 3.0.5419.0

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	Type	Description	Required
Codepage	String	The code page integer value or web name for the resulting database. 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 [Code Pages](#) 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](#)

Version 3.0.5419.0

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](#)
- [ComPlusAssemblyDependency](#)
- [ComPlusComponent](#)
- [ComPlusGroupInApplicationRole](#)
- [ComPlusGroupInPartitionRole](#)
- [ComPlusInterface](#)
- [ComPlusMethod](#)
- [ComPlusPartition](#)
- [ComPlusPartitionRole](#)
- [ComPlusPartitionUser](#)
- [ComPlusRoleForComponent](#)
- [ComPlusRoleForInterface](#)
- [ComPlusRoleForMethod](#)

- [ComPlusSubscription](#)
- [ComPlusUserInApplicationRole](#)
- [ComPlusUserInPartitionRole](#)

ComPlusApplication Element (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 locator, referencing an existing application.

If the element is a child of a ComPlusPartition element, or have its Partition attribute set, the application will be installed under the referenced partition.

Windows Installer references

None

Parents

[ComPlusPartition](#), [Component](#), [Fragment](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

1. Choice of elements (min: 0, max: unbounded)
 - [ComPlusApplicationRole](#) (min: 0, max: unbounded)
 - [ComPlusAssembly](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
Id	String	Identifier for the element
AccessChecksLevel	Enumeration	This attribute's value will be one of the following <i>applicationLevel</i>

applicationCompo

Activation	Enumeration	This attribute's value can be one of the following: <i>inproc</i> <i>local</i>
ApplicationAccessChecksEnabled	YesNoType	
ApplicationDirectory	String	
ApplicationId	String	Id for the application. This attribute can be optional. In which case an id will be generated on instantiation. If the element is a localizable element, this attribute can be optional. If the value is provided for the Name attribute.
Authentication	Enumeration	This attribute's value can be one of the following: <i>default</i> <i>none</i> <i>connect</i> <i>call</i> <i>packet</i> <i>integrity</i> <i>privacy</i>
AuthenticationCapability	Enumeration	This attribute's value can be one of the following: <i>none</i> <i>secureReference</i> <i>staticCloaking</i> <i>dynamicCloaking</i>

Changeable	YesNoType	
CommandLine	String	
ConcurrentApps	Int	
CreatedBy	String	
CRMEnabled	YesNoType	
CRMLogFile	String	
Deleteable	YesNoType	
Description	String	
DumpEnabled	YesNoType	
DumpOnException	YesNoType	
DumpOnFailfast	YesNoType	
DumpPath	String	
EventsEnabled	YesNoType	
Identity	String	
ImpersonationLevel	Enumeration	This attribute's value be one of the following <i>anonymous</i> <i>identify</i> <i>impersonate</i> <i>delegate</i>
IsEnabled	YesNoType	
MaxDumpCount	Int	
Name	String	Name of the application. This attribute can be omitted if the element has a locator, and a value is provided for the Partition attribute.
Partition	String	If the element is not a ComPlusPartition element, this attribute be provided with the ComPlusPartition attribute.

		representing the p the application bel
Password	String	
QCAuthenticateMsgs	Enumeration	This attribute's val be one of the follo <i>secureApps</i> <i>off</i> <i>on</i>
QCListenerMaxThreads	Int	
QueueListenerEnabled	YesNoType	
QueuingEnabled	YesNoType	
RecycleActivationLimit	Int	
RecycleCallLimit	Int	
RecycleExpirationTimeout	Int	
RecycleLifetimeLimit	Int	
RecycleMemoryLimit	Int	
Replicable	YesNoType	
RunForever	YesNoType	
ShutdownAfter	Int	
SoapActivated	YesNoType	
SoapBaseUrl	String	
SoapMailTo	String	
SoapVRoot	String	
SRPEnabled	YesNoType	
SRPTrustLevel	Enumeration	This attribute's val be one of the follo <i>disallowed</i> <i>fullyTrusted</i>
ThreeGigSupportEnabled	YesNoType	

See Also

[Complus Schema](#)

Version 3.0.5419.0

ComPlusApplicationRole 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 locator, referencing an existing application role.

Windows Installer references

None

Parents

[ComPlusApplication](#), [Component](#), [Fragment](#), [Module](#), [Product](#)

Inner Text

None

Children

Sequence (min: 1, max: 1)

- Choice of elements (min: 0, max: unbounded)
 - [ComPlusGroupInApplicationRole](#) (min: 0, max: unbounded)
 - [ComPlusUserInApplicationRole](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Application	String	If the element is not a child of a ComPlusApplication element, this attribute should be provided with the id of a ComPlusApplication element representing the application the	

role belongs to.

Description	String		
Name	String	Name of the application role.	Yes

See Also

[Complus Schema](#)

Version 3.0.5419.0

ComPlusAssembly Element (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)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Application	String	If the element is not a child of a ComPlusApplication element, this attribute should be provided	

with the id of a ComPlusApplication element representing the application the assembly is to be registered in. This attribute can be omitted for a .NET assembly even if the application is not a child of a ComPlusApplication element.

AssemblyName	String	The name of the assembly used to identify the assembly in the GAC. This attribute can be provided only if DllPathFromGAC is set to "yes".
DllPath	String	The path to locate the assembly DLL during registration. This attribute should be provided if DllPathFromGAC is not set to "yes".
DllPathFromGAC	YesNoType	Indicates that the DLL path should be extracted from the GAC instead for being provided in the DllPath attribute. If this attribute is set to "yes", the name of the assembly can be provided using the

AssemblyName attribute. Or, if this AssemblyName attribute is missing, the name will be extracted from the MsiAssemblyName table using the id of the parent Component element.

EventClass	YesNoType	Indicates that the assembly is to be installed as an event class DLL. This attribute is only valid for native assemblies. The assembly will be installed with the COM+ catalog's InstallEventClass() function.
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: <i>native</i> <i>.net</i>	Yes

Remarks

When installing a native assembly, all components contained in the assembly must be represented as ComPlusComponent elements under this element. Any component not listed will not be removed during uninstall.

The fields DllPath, TlbPath and PSDllPath are formatted fields that should contain file paths to their respective file types. A typical value for DllPath for example, should be something like “[#MyAssembly_dll]”, where “MyAssembly_dll” is the key of the dll file in the File table.

Warning: The assembly name provided in the AssemblyName attribute must be a fully specified assembly name, if a partial name is provided a random assembly matching the partial name will be selected.

See Also

[Complus Schema](#)

Version 3.0.5419.0

ComPlusAssemblyDependency Element (Complus Extension)

Description

Defines a dependency between two assemblies. This element affects the order in which assemblies are registered. Any assemblies referenced by this element are guaranteed to be registered before, and unregistered after, the assembly referenced by the parent ComPlusAssembly element.

Windows Installer references

None

Parents

[ComPlusAssembly](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
RequiredAssembly	String	Reference to the id of the assembly required by the parent ComPlusAssembly element.	Yes

Remarks

It is only necessary to explicitly specify dependencies between assemblies contained in the same package (MSI or MSM). Assemblies merged in to a package from a merge module will always be installed before any assemblies specified in the base package. Assemblies merged in from different merge modules are sequenced using the ModuleDependency MSI table. It is not possible to have cross dependencies between merge modules

or have an assembly in a merge module depend on an assembly in the base package.

See Also

[Complus Schema](#)

Version 3.0.5419.0

ComPlusComponent 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)
 - [ComPlusRoleForComponent](#) (min: 0, max: unbounded)
 - [ComPlusSubscription](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description
Id	String	Identifier for the element.
AllowInprocSubscribers	YesNoType	
CLSID	Uuid	CLSID of the component.
ComponentAccessChecksEnabled	YesNoType	
ComponentTransactionTimeout	Int	
ComponentTransactionTimeoutEnabled	YesNoType	
COMTIIntrinsics	YesNoType	
ConstructionEnabled	YesNoType	

ConstructorString	String	
CreationTimeout	Int	
Description	String	
EventTrackingEnabled	YesNoType	
ExceptionClass	String	
FireInParallel	YesNoType	
IISIntrinsics	YesNoType	
InitializesServerApplication	YesNoType	
IsEnabled	YesNoType	
IsPrivateComponent	YesNoType	
JustInTimeActivation	YesNoType	
LoadBalancingSupported	YesNoType	
MaxPoolSize	Int	
MinPoolSize	Int	
MultInterfacePublisherFilterCLSID	String	
MustRunInClientContext	YesNoType	
MustRunInDefaultContext	YesNoType	
ObjectPoolingEnabled	YesNoType	
PublisherID	String	
SoapAssemblyName	String	
SoapTypeName	String	
Synchronization	Enumeration	This attribute value must be one of the following: <i>ignored</i> <i>none</i> <i>supported</i> <i>required</i> <i>requiresNew</i>
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: <i>any</i> <i>readUnComr</i> <i>readCommitt</i> <i>repeatableRe</i> <i>serializable</i>
------------------	-------------	--

See Also
[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 locator element. To reference a locator element use the ApplicationRole attribute.

Windows Installer references

None

Parents

[ComPlusApplicationRole](#), [Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	If the element is not a child of a ComPlusApplicationRole element, this attribute should be provided with the id of a ComPlusApplicationRole element representing the application role the user is to be added to.	
Group	String	Foreign key into the Group	Yes

table.

See Also

[Complus Schema](#)

Version 3.0.5419.0

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	Type	Description	Required
Id	String	Identifier for the element.	Yes
Group	String	Foreign key into the Group table.	Yes
PartitionRole	String	The id of a ComPlusPartitionRole element representing the partition the user should be added to.	

See Also

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

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Description	String		
IID	Uuid	IID of the interface.	Yes
QueuingEnabled	YesNoType		

See Also

[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)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
AutoComplete	YesNoType		
Description	String		
Index	Int	Dispatch id of the method. If this attribute is not set a value must be provided for the Name attribute.	
Name	String	Name of the method. If this attribute is not set a value must be provided for the Index attribute.	

See Also

[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 locator, referencing an existing partition.

Windows Installer references

None

Parents

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

Inner Text

None

Children

Sequence (min: 1, max: 1)

- Choice of elements (min: 0, max: unbounded)
 - [ComPlusApplication](#) (min: 0, max: unbounded)
 - [ComPlusPartitionRole](#) (min: 0, max: unbounded)
 - [ComPlusPartitionUser](#) (min: 0, max: unbounded)

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
Changeable	YesNoType		
Deleteable	YesNoType		
Description	String		
Name	String	Name of the partition. This attribute can be omitted if the element is a locator, and	

		a value is provided for the PartitionId attribute.
PartitionId	String	Id for the partition. This attribute can be omitted, in which case an id will be generated on install. If the element is a locator, this attribute can be omitted if a value is provided for the Name attribute.

See Also

[Complus Schema](#)

Version 3.0.5419.0

ComPlusPartitionRole Element (Complus Extension)

Description

Defines a COM+ partition role. Partition roles can not be created; this element can only be used as a locator to reference an existing role.

Windows Installer references

None

Parents

[ComPlusPartition](#), [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	Type	Description	Required
Id	String	Identifier for the element.	Yes
Name	String	Name of the partition role.	Yes
Partition	String	The id of a ComPlusPartition element representing the partition the role belongs to.	

See Also

[Complus Schema](#)

ComPlusPartitionUser 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	Type	Description	Required
Id	String	Identifier for the element.	Yes
Partition	String	The id of a ComPlusPartition element representing the partition that will be the default partition for the user.	
User	String	Foreign key into the User table.	Yes

See Also

[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](#)

Version 3.0.5419.0

ComPlusRoleForInterface 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](#)

Version 3.0.5419.0

ComPlusRoleForMethod 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](#)

Version 3.0.5419.0

ComPlusSubscription 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	Type	Description	Required
Id	String	Identifier for the element.	Yes
Component	String	If the element is not a child of a ComPlusComponent element, this attribute should be provided with the id of a ComPlusComponent element representing the component the subscription is to be created for.	
Description	String		

Enabled	YesNoType		
EventClassPartitionID	String		
EventCLSID	String	CLSID of the event class for the subscription. If a value for this attribute is not provided, a value for the PublisherID attribute must be provided.	
FilterCriteria	String		
InterfaceID	String		
MachineName	String		
MethodName	String		
Name	String	Name of the subscription.	Yes
PerUser	YesNoType		
PublisherID	String	Publisher id for the subscription. If a value for this attribute is not provided, a value for the EventCLSID attribute must be provided.	
Queued	YesNoType		
SubscriberMoniker	String		
SubscriptionId	String	Id of the subscription. If a value is not provided for this attribute, an id will be generated during installation.	
UserName	String		

See Also
[Complus Schema](#)

Version 3.0.5419.0

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 locator element. To reference a locator element use the ApplicationRole attribute.

Windows Installer references

None

Parents

[ComPlusApplicationRole](#), [Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the element.	Yes
ApplicationRole	String	If the element is not a child of a ComPlusApplicationRole element, this attribute should be provided with the id of a ComPlusApplicationRole element representing the application role the user is to be added to.	
User	String	Foreign key into the User table.	Yes

See Also
[Complus Schema](#)

Version 3.0.5419.0

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	Type	Description	Required
Id	String	Identifier for the element.	Yes
PartitionRole	String	The id of a ComPlusPartitionRole element representing the partition the user should be added to.	
User	String	Foreign key into the User table.	Yes

See Also

[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]{4}\-?[0-9A-Fa-f]{12}[])]?|[(?)\{8}\-?\{4}\-?\{4}\-?\{4}\-?\{12}\}[])]?|PUT\-\GUID\-(?:\d+\-)?HERE|([!\$])\(\(var|\(loc|\(wix)\.[_A-Za-z][0-9A-Za-z_]*\)|\)*'.

See Also

[Complus 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

[Complus Schema](#)

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

[Complus 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

[Complus 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Complus Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Complus Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Complus Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Complus 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: `'[^\\?|><:\^*" \+,;=\[\]\\.]{1,8}(\. [^\\?|><:\^*" \+,;=\[\]\\.]{0,3})?(![!$])\(\loc\.[_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Complus Schema](#)

uuid (Simple Type)

Description

Values of this type will look like: "01234567-89AB-CDEF-0123-456789ABCDEF".

Pattern Type

Must match the regular expression: '[0-9A-Fa-f]{8}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{4}\-[0-9A-Fa-f]{12}'.

See Also

[Complus 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

[Complus Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Complus Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: `'[^\\|><:/"+,;=\\[\.]{1,16}(\. [^\\|><:/"+,;=\\[\.]{0,6})?(!$)\\(loc\\. [A-Za-z][0-9A-Za-z_]*\\)'`

See Also

[Complus 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

[Complus Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Complus Schema](#)

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	Type	Description
AddRemovePrograms	YesNoType	Specifies that the DIFxApp CustomAction entry in the Add/Remove Programs table. The default is 'yes'.
DeleteFiles	YesNoType	If set to "yes", configures DIFxApp to delete files that were copied to the system folder when a driver package was installed. If set to "no" or not present, DIFxApp does not delete files from a system. Note that configuration of these files is controlled by the FileComponent component that represents the FileComponent MsiDriverPackages custom table. Setting DriverDeleteFiles to "yes" sets the DriverDeleteFiles entry value. Setting DriverDeleteFiles to "no" or not present, DIFxApp uses a default value.

ForceInstall	YesNoType	Specifies that the DIFxApp CustomAction will force the installation of a new Plug and Play driver even if the currently installed driver is a better match than the new driver. This is an excellent way to ensure the DIFxApp will recognize the Component containing the driver. The default is null which means DIFxApp will install a driver via DIFxApp CustomAction. For more information, see http://www.microsoft.com/whdc/driver/forceinstall/ .
Legacy	YesNoType	If set to "yes", configures DIFxApp to install driver packages and driver packages that are not pre-signed. For more information, see "Installing Driver Packages in Legacy Mode" early in the installation. The default attribute is set to "no" or not present. No DIFxApp will install unsigned driver packages. No Flags entry value of the component. Setting DriverLegacy to "yes" sets the Flags entry value. Setting DriverLegacy to "no" sets the bit in the Flags entry value to 0. If the bit is set, DIFxApp will install unsigned driver packages if they are present, DIFxApp uses a default Flags entry value of 0.
PlugAndPlayPrompt	YesNoType	Specifies that the DIFxApp CustomAction will prompt the user to connect the Plug and Play driver if it is not connected. The default is 'yes'.
Sequence	Integer	Specifies an optional installation order for DIFxApp CustomActions. The installation package in the order of the sequence numbers. The same sequence number can be used for more than one driver; however, the order of installation packages with the same sequence number cannot be determined.

See Also

[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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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](#)

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](#)

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

[Difxapp 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

[Difxapp 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Difxapp Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Difxapp Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Difxapp Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Difxapp 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: `'[^\|>:<:\^*" \+,;=\[\] \.]{1,8}(\. [^\|>:<:\^*" \+,;=\[\] \.]{0,3})?(!\$)\(loc\[_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Difxapp 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

[Difxapp Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Difxapp Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: `'[^\\|><:/"+,;=\\[\.]{1,16}(\. [^\\|><:/"+,;=\\[\.]{0,6})?(!\$)\\(loc\\. [A-Za-z][0-9A-Za-z_]*\\)'`

See Also

[Difxapp 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

[Difxapp Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Difxapp Schema](#)

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](#)

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	Type	Description	Required
Id	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.	
Program	String	Path to a target program to be granted access to all incoming ports and protocols. Note that this is a formatted field, so you can use [#fileId] syntax to refer to a file being installed. If you use Program, you cannot also use File, Port, or Protocol.	
Protocol	Enumeration	IP protocol used for this firewall exception. If not specified, "tcp" is assumed. If you use Protocol, you must also specify Port and you cannot also use File or Program. This attribute's value must be one of the following: <i>tcp</i> <i>udp</i>	
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 3.0.5419.0

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](#)

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Firewall 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

[Firewall Schema](#)

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

[Firewall 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

[Firewall 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Firewall Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Firewall Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Firewall Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Firewall 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: `'[^\\?|><:\^*" \+,;=\[\] \.]{1,8}(\. [^\\?|><:\^*" \+,;=\[\] \.]{0,3})?(![!$])\(\loc\. [_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Firewall 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

[Firewall Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Firewall Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: '[^\\|><:"'+,;=\\[\.]{1,16}\\. [^\\|><:"'+,;=\\[\.]{0,6}?|(!\$)\\(loc\\.[_A-Za-z][0-9A-Za-z_\\.]*\\)'

See Also

[Firewall 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

[Firewall Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Firewall Schema](#)

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](#)

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	Type	Description	Required
Id	Guid	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](#)

Version 3.0.5419.0

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 [Game Explorer Tasks](#). 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	Type	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

[Gaming Schema](#)

SupportTask Element (Gaming Extension)

Description

Creates an Internet shortcut and registers it as a "support task" in Game Explorer. For more information, see [Game Explorer Tasks](#). 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	Type	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

[Gaming Schema](#)

IsRichSavedGame Attribute (Gaming Extension)

Description

Registers this extension for the [rich saved games](#) property handler on Windows Vista and later.

Windows Installer references

None

Parents

[Extension](#)

See Also

[Gaming Schema](#)

Version 3.0.5419.0

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Gaming 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

[Gaming Schema](#)

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

[Gaming 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

[Gaming 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Gaming Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Gaming Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Gaming Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Gaming 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: `'[^\|>:<:\^*" \+,;=\[\] \.]{1,8}(\. [^\|>:<:\^*" \+,;=\[\] \.]{0,3})?(![!$])\ (loc\.[_A-Za-z][0-9A-Za-z_\.]*)'`

See Also

[Gaming 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

[Gaming Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Gaming Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: '[^\\|><:"'+,;=\\[\.]{1,16}\\. [^\\|><:"'+,;=\\[\.]{0,6}?|(!\$)\\(loc\\. [_A-Za-z][0-9A-Za-z_\\.]*\\)'

See Also

[Gaming 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

[Gaming Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Gaming Schema](#)

Iis 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](#)
- [HttpHeader](#)
- [MimeMap](#)
- [RecycleTime](#)
- [WebAddress](#)
- [WebApplication](#)
- [WebApplicationExtension](#)
- [WebAppPool](#)
- [WebDir](#)
- [WebDirProperties](#)
- [WebError](#)
- [WebFilter](#)
- [WebLog](#)
- [WebProperty](#)
- [WebServiceExtension](#)

- [WebSite](#)
- [WebVirtualDir](#)

Certificate Element (Iis Extension)

Description

Used to install and uninstall certificates.

Windows Installer references

None

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

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

the certificate authority to request the certificate from. This attribute may be set via a formatted Property (e.g. [MyProperty]).

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

StoreName	Enumeration	This attribute's value must be one of the following: <i>ca</i> 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. <i>my</i> Use the "personal" value instead. <i>personal</i> Contains personal certificates. These certificates will usually have an associated private key. This store is often referred to as the "MY" certificate store. <i>request</i> <i>root</i> Contains the certificates of certificate authorities that the	Yes
-----------	-------------	---	-----

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

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

Version 3.0.5419.0

CertificateRef Element (Iis 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

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

HTTPHeader Element (Iis 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	Type	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

[Iis Schema](#)

Version 3.0.5419.0

MimeMap Element (IIS Extension)

Description

MimeMap definition for IIS resources.

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

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

See Also

[IIS Schema](#)

RecycleTime Element (Iis Extension)

Description

IIS6 Application Pool Recycle Times on 24 hour clock.

Windows Installer references

None

Parents

[WebAppPool](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Value	String	Pattern: '\d{1,2}:\d{2}'.	Yes

See Also

[Iis Schema](#)

WebAddress Element (Iis Extension)

Description

WebAddress for WebSite

Windows Installer references

None

Parents

[WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
Header	String		
IP	String	<p>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:</p> <ul style="list-style-type: none">• 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 (Iis 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)

Attributes

Name	Type	Description	Required
Id	String		Yes
AllowSessions	YesNoDefaultType	Sets the Enable Session State option. When enabled, you can set the session timeout using the SessionTimeout attribute.	
Buffer	YesNoDefaultType	Sets the option that enables response	

buffering in the application, which allows ASP script to set response headers anywhere in the script.

ClientDebugging	YesNoDefaultType	Enable ASP client-side script debugging.
DefaultScript	Enumeration	Sets the default script language for the site. This attribute's value must be one of the following: <i>VBScript</i> <i>JScript</i>
Isolation	Enumeration	Sets the application isolation level for this application for pre-IIS 6 applications. This attribute's value must be one of the following: <i>low</i> Means the application executes within 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
[IIS Schema](#)

Version 3.0.5419.0

WebApplicationExtension Element (Iis Extension)

Description

Extension for WebApplication

Windows Installer references

None

Parents

[WebApplication](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
CheckPath	YesNoType		
Executable	String	usually a Property that resolves to short file name path	Yes
Extension	String	Extension being registered. Do not prefix with a '.' (e.g. you should use "html", not ".html"). To register for all extensions, use Extension="*". To register a wildcard application map (which handles all requests, even those for directories or files with no extension) omit the Extension attribute completely.	

Script	YesNoType
--------	---------------------------

Verbs	String
-------	--------

See Also

[Iis Schema](#)

Version 3.0.5419.0

WebAppPool Element (Iis 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)

Attributes

Name	Type	Description	Required
Id	String	Id of the AppPool.	Yes
CpuAction	Enumeration	Action taken when CPU exceeds maximum CPU use (as defined with MaxCpuUsage and RefreshCpu). This attribute's value must be one of the following: <i>none</i>	

shutdown

Identity	Enumeration	Identity you want the AppPool to run under. Use the 'other' value in conjunction with the User attribute to specify non-standard user. This attribute's value must be one of the following: <i>networkService</i> <i>localService</i> <i>localSystem</i> <i>other</i>	
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
[lis Schema](#)

Version 3.0.5419.0

WebDir Element (Iis 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)

Attributes

Name	Type	Description	Required
Id	String		Yes
DirProperties	String	References the Id attribute for a WebDirProperties element that specifies the security and access properties for this web directory. 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
[lis Schema](#)

Version 3.0.5419.0

WebDirProperties Element (Iis 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

Attributes

Name	Type	Description
Id	String	
AccessSSL	YesNoType	A value of true indicates that file access requires SSL file permission processing, with or without a client certificate. This corresponds to the AccessSSL flag for the AccessSSLFlags IIS metabase property.

AccessSSL128	YesNoType	A value of true indicates that file access requires SSL file permission processing with a minimum key size of 128 bits, with or without a client certificate. This corresponds to the AccessSSL128 flag in the AccessSSLFlags IIS metabase property.
AccessSSLMapCert	YesNoType	This corresponds to the AccessSSLMapCert flag for the AccessSSLFlags IIS metabase property.
AccessSSLNegotiateCert	YesNoType	This corresponds to the AccessSSLNegotiateCert flag for the AccessSSLFlags IIS metabase property.
AccessSSLRequireCert	YesNoType	This corresponds to the AccessSSLRequireCert flag for the AccessSSLFlags IIS metabase property.
AnonymousAccess	YesNoType	Sets the Enable Anonymous Access checkbox, which makes anonymous users to Windows user accounts. When setting this to true, you should also provide the user account used as the AnonymousUser attribute, and determine what setting to use for the IISControlledPassword attribute. Defaults to false.
AnonymousUser	String	Reference to the Id

attribute on the Use element to be used the anonymous use the directory. See the User element for more information.

AspDetailedError	YesNoType	Sets the option for whether to send detailed ASP errors back to the client on script error. Default is 'no.'
AuthenticationProviders	String	Comma delimited list in order of precedence of Windows authentication providers that IIS will attempt to use: NTLM, Kerberos, Negotiate, and others.
BasicAuthentication	YesNoType	Sets the Basic Authentication option which allows clients to provide credentials in plaintext over the wire. Defaults to 'no.'
CacheControlCustom	String	Custom HTTP 1.1 cache control directives.
CacheControlMaxAge	NonNegativeInteger	Integer value specifying the cache control maximum age value.
ClearCustomError	YesNoType	Specifies whether IIS will return custom errors from this directory.
DefaultDocuments	String	The list of default documents to set for the web directory, in comma delimited format.
DigestAuthentication	YesNoType	Sets the Digest

		Authentication option which allows using (authentication with domain user account Defaults to 'no.'
Execute	YesNoType	
HttpExpires	String	Value to set the HttpExpires attribute for a Web Dir in the metabase.
IIsControlledPassword	YesNoType	Sets whether IIS should control the password used for the Windows account specified in AnonymousUser attribute. Defaults to
Index	YesNoType	Sets the Index Resc option, which specifies whether this web directory should be indexed. Defaults to
LogVisits	YesNoType	Sets whether visits to site should be logged Defaults to 'no.'
PassportAuthentication	YesNoType	Sets the Passport Authentication option which allows clients provide credentials to .Net Passport account Defaults to 'no.'
Read	YesNoType	
Script	YesNoType	
WindowsAuthentication	YesNoType	Sets the Windows Authentication option which enables integrated Windows authentication to be used on the site

Defaults to 'no.'

Write

[YesNoType](#)

See Also

[Iis Schema](#)

Version 3.0.5419.0

WebError Element (IIS 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	Type	Description	Required
ErrorCode	Integer	HTTP 1.1 error code.	Yes
File	String	File to be sent to the client for this error code and sub code. This can be formatted. For example: [#FileId].	
SubCode	Integer	Error sub code. Set to 0 to get the wild card "*".	Yes
URL	String	URL to be sent to the client for this error code and sub code. This can be formatted.	

Remarks

You can only use error code and sub code combinations which are supported by IIS. Attempting to set a custom error for an error code and sub code combination that is not supported by IIS (in the default list of error codes) will result in an installation failure.

See Also

[IIS Schema](#)

Version 3.0.5419.0

WebFilter Element (IIS Extension)

Description

IIS Filter for a Component

Windows Installer references

None

Parents

[Component](#), [WebSite](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	The unique Id for the web filter.	Yes
Description	String	Description of the filter.	
Flags	Integer	Sets the MD_FILTER_FLAGS metabase key for the filter. This must be an integer. See MSDN 'FilterFlags' documentation for more details.	
LoadOrder	String	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 file. This should usually be a value like '[!FileId]', where 'FileId' is the file identifier of the filter	Yes

		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

[Iis Schema](#)

Version 3.0.5419.0

WebLog Element (Iis Extension)

Description

WebLog definition.

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Identifier for the WebLog.	Yes
Type	Enumeration	This attribute's value must be one of the following: <i>IIS</i> Microsoft IIS Log File Format <i>NCSA</i> NCSA Common Log File Format <i>none</i> Disables logging. <i>ODBC</i> ODBC Logging <i>W3C</i> W3C Extended Log File Format	Yes

See Also
[IIS Schema](#)

Version 3.0.5419.0

WebProperty Element (Iis Extension)

Description

IIS Properties

Windows Installer references

None

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	Enumeration	This attribute's value must be one of the following: <i>ETagChangeNumber</i> <i>IIs5IsolationMode</i> <i>MaxGlobalBandwidth</i> <i>LogInUTF8</i>	Yes
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

[IIS Schema](#)

Version 3.0.5419.0

WebServiceExtension Element (Iis 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

Attributes

Name	Type	Description	Required
Id	String		Yes
Allow	YesNoType	Indicates if the extension is allowed or denied.	Yes
Description	String	Description of the extension.	
File	String	Usually a Property that resolves to short file name path	Yes
Group	String	String used to identify groups of extensions.	
UIDeletable	YesNoType	Indicates if the UI is allowed to delete the extension from the list of not. Default: Not deletable.	

See Also
[lis Schema](#)

Version 3.0.5419.0

WebSite Element (Iis 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)

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

Attributes

Name	Type	Description	Required
Id	String	Identifier for the WebSite. Used within the MSI package only.	Yes
AutoStart	YesNoType	Specifies whether to automatically	

		start the web site.	
ConfigureIfExists	YesNoType	Specifies whether to configure the web site if it already exists. Note: This will not affect uninstall behavior. If the web site exists on uninstall, it will be removed.	
ConnectionTimeout	NonNegativeInteger	Sets the timeout value for connections in seconds.	
Description	String	This is the name of the web site that will show up in the IIS management console.	Yes
Directory	String	Root directory of the web site. Resolved to a directory in the Directory table at install time by the server custom actions.	
DirProperties	String	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.
Siteld	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

[IIS Schema](#)

WebVirtualDir Element (IIS 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)

Attributes

Name	Type	Description	Required
Id	String		Yes
Alias	String	Sets the application name, which is the URL relative path used to access this virtual directory	Yes
Directory	String	References the Id attribute for a Directory element that points to the content for this	Yes

		virtual directory.
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
[IIS Schema](#)

Version 3.0.5419.0

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]{4}\-?\[0-9A-Fa-f]{12}\])]?|[(\?[0-9A-Fa-f]{8}\-?\[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](#)

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

[lis Schema](#)

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

[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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[lis Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[lis Schema](#)

PercentType (Simple Type)

Description

Values of this type are any integers between 0 and 100, inclusive.

xs:nonNegativeInteger Type

- xs:maxInclusive value='100'

See Also

[lis Schema](#)

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](#)

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: `'[^\|>:<:\^*" \+,;=\[\] \.]{1,8}(\. [^\|>:<:\^*" \+,;=\[\] \.]{0,3})?(![!$])\ (loc\.[_A-Za-z][0-9A-Za-z_\.]*)'`

See Also

[IIS 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

[lis Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[IIS Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: `'[^\|><:/"+,;=\\[\.]{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

[lis Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[lis Schema](#)

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"?>
 <IsolatedApp
 xmlns="http://wix.sourceforge.net/schemas/clickthrough/isolatedapp/2006"
 .
 .
 .
 </IsolatedApp>
```

# Application Element (IsolatedApp Extension)

## Description

None

## Windows Installer references

None

## Parents

[IsolatedApp](#)

## Inner Text

None

## Children

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

- [Details](#) (min: 0, max: 1)
- [EntryPoint](#) (min: 0, max: 1)
- [Icon](#) (min: 0, max: 1)
- [Id](#) (min: 0, max: 1)
- [Name](#) (min: 0, max: 1)
- [Source](#) (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](#)

*Version 3.0.5419.0*

# 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](#)

*Version 3.0.5419.0*

# 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           | Type    | Description | Required |
|----------------|---------|-------------|----------|
| PackageVersion | Boolean |             |          |

## See Also

[IsolatedApp Schema](#)

*Version 3.0.5419.0*



# 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](#)

*Version 3.0.5419.0*

# 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  | Type    | Description | Required |
|-------|---------|-------------|----------|
| Index | Integer |             |          |

## See Also

[IsolatedApp Schema](#)

Version 3.0.5419.0

# 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](#)

*Version 3.0.5419.0*

# 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](#)

*Version 3.0.5419.0*

# 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](#)

*Version 3.0.5419.0*

# 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](#)

*Version 3.0.5419.0*

# 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)
- [Icon](#) (min: 0, max: 1)
- [Id](#) (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](#)

*Version 3.0.5419.0*



# 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](#)

*Version 3.0.5419.0*

# 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 3.0.5419.0*

# 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](#)

*Version 3.0.5419.0*

# 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]{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](#)

# 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](#)

# 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             | Type                      | Description                                                                            | Required |
|------------------|---------------------------|----------------------------------------------------------------------------------------|----------|
| Id               | String                    |                                                                                        | Yes      |
| Authenticate     | <a href="#">YesNoType</a> | Default: No.                                                                           |          |
| BasePriority     | Integer                   |                                                                                        |          |
| Journal          | <a href="#">YesNoType</a> | Default: No.                                                                           |          |
| JournalQuota     | Integer                   |                                                                                        |          |
| Label            | String                    |                                                                                        | Yes      |
| MulticastAddress | String                    |                                                                                        |          |
| PathName         | String                    |                                                                                        | Yes      |
| PrivLevel        | Enumeration               | This attribute's value must be one of the following:<br><i>none</i><br><i>optional</i> |          |

*body*

|                 |                           |              |  |
|-----------------|---------------------------|--------------|--|
| Quota           | Integer                   |              |  |
| ServiceTypeGuid | String                    |              |  |
| Transactional   | <a href="#">YesNoType</a> | Default: No. |  |

**See Also**

[Msmq Schema](#)

*Version 3.0.5419.0*



# MessageQueuePermission Element (Msmq Extension)

## Description

None

## Windows Installer references

None

## Parents

[Component](#), [MessageQueue](#)

## Inner Text

None

## Children

None

## Attributes

| Name                   | Type                      | Description | Required |
|------------------------|---------------------------|-------------|----------|
| Id                     | String                    |             | Yes      |
| ChangeQueuePermissions | <a href="#">YesNoType</a> |             |          |
| DeleteJournalMessage   | <a href="#">YesNoType</a> |             |          |
| DeleteMessage          | <a href="#">YesNoType</a> |             |          |
| DeleteQueue            | <a href="#">YesNoType</a> |             |          |
| GetQueuePermissions    | <a href="#">YesNoType</a> |             |          |
| GetQueueProperties     | <a href="#">YesNoType</a> |             |          |
| Group                  | String                    |             |          |
| MessageQueue           | String                    |             |          |
| PeekMessage            | <a href="#">YesNoType</a> |             |          |
| QueueGenericAll        | <a href="#">YesNoType</a> |             |          |
| QueueGenericExecute    | <a href="#">YesNoType</a> |             |          |
| QueueGenericRead       | <a href="#">YesNoType</a> |             |          |
| QueueGenericWrite      | <a href="#">YesNoType</a> |             |          |

|                       |                           |  |
|-----------------------|---------------------------|--|
| ReceiveJournalMessage | <a href="#">YesNoType</a> |  |
| ReceiveMessage        | <a href="#">YesNoType</a> |  |
| SetQueueProperties    | <a href="#">YesNoType</a> |  |
| TakeQueueOwnership    | <a href="#">YesNoType</a> |  |
| User                  | String                    |  |
| WriteMessage          | <a href="#">YesNoType</a> |  |

**See Also**

[Msmq Schema](#)

*Version 3.0.5419.0*

# 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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Msmq 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

[Msmq Schema](#)

# 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

[Msmq 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

[Msmq 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]\*|(![\$])\((?:loc|bind)\.[\_A-Za-z][0-9A-Za-z\_\.]+\)'.

## See Also

[Msmq Schema](#)

# LongFileNameType (Simple Type)

## Description

Values of this type will look like: "Long File Name.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

[Msmq Schema](#)



# PatchClassificationType (Simple Type)

## Description

Category of update.

## Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

## See Also

[Msmq Schema](#)

# RegistryRootType (Simple Type)

## Description

Values of this type represent possible registry roots.

## Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

## See Also

[Msmq 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: `'[^\\?|><:\^*" \+,;=\[\] \. ]{1,8}(\. [^\\?|><:\^*" \+,;=\[\] \. ]{0,3})?(![!$])\(\loc\.[_A-Za-z][0-9A-Za-z_\.]*\)'`.

## See Also

[Msmq 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

[Msmq Schema](#)

# WildcardLongFileNameType (Simple Type)

## Description

Values of this type will look like: "Long File Name.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

[Msmq Schema](#)

# WildcardShortFileNameType (Simple Type)

## Description

Values of this type will look like: "File?.\*". 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: `'[^\|><:"'+,;=\\]\\. ]{1,16}(\. [^\|><:"'+,;=\\]\\. ]{0,6})?(!\$)\(loc\.[_A-Za-z][0-9A-Za-z_]*\)'`.

## See Also

[Msmq 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

[Msmq Schema](#)

# YesNoType (Simple Type)

## Description

Values of this type will either be "yes" or "no".

## Enumeration Type

Possible values: {no, yes}

## See Also

[Msmq Schema](#)



# 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

- [NativeImage](#)

# NativeImage 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             | Type   | Description                                                                                                                                                                                                                                                                                                                                                          | Rec |
|------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Id               | String | The identifier for this NativeImage.                                                                                                                                                                                                                                                                                                                                 | Yes |
| AppBaseDirectory | String | The identifier of the directory to use for locating dependent assemblies. For DLL assemblies and assemblies installed to the 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: specifying File/@AssemblyApplication will force an assembly to install to a private location next to the indicated application, whereas this AssemblyApplication attribute will be used to help resolve dependent assemblies while generating native images for this assembly).

|              |                           |                                                                                                                                                                                                                             |
|--------------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Debug        | <a href="#">YesNoType</a> | Set to "yes" to generate native images that can be used under a debugger. The default value is "no".                                                                                                                        |
| Dependencies | <a href="#">YesNoType</a> | 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:<br><i>32bit</i><br>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:<br><i>0</i><br>This is the highest priority, it means that image generation occurs synchronously during the setup process. This option will slow down setup performance.<br><i>1</i><br>This will queue image generation to the NGen service to occur immediately. This option will slow down setup performance.<br><i>2</i><br>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.<br><i>3</i><br>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 | <a href="#">YesNoType</a> | 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"?>
<OfficeAddin
xmlns="http://wix.sourceforge.net/schemas/clickthrough/officeaddin/2006"
.
.
.
</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)
- [Icon](#) (min: 0, max: 1)
- [Id](#) (min: 0, max: 1)
- [Name](#) (min: 0, max: 1)
- [Source](#) (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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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	Type	Description	Required
Index	Integer		

See Also

[OfficeAddin Schema](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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)
- [Icon](#) (min: 0, max: 1)
- [Id](#) (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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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](#)

Version 3.0.5419.0

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 3.0.5419.0

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](#)

Version 3.0.5419.0

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]{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](#)
- [SnapIn](#)
- [TypesFile](#)

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](#), [SnapIn](#)

Inner Text

None

Children

None

Attributes

Name	Type	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	Type	Description	Rec
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 snap-in. 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-

VendorIndirect	EmbeddedResource	in vendor. 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

[PowerShell](#) 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

[Ps Schema](#)

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	Type	Description	Required
FileId	String	Reference to the types File ID. This is required when nested under the SnapIn element.	
SnapIn	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

[Ps Schema](#)

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 3.0.5419.0

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Ps 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

[Ps Schema](#)

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

[Ps Schema](#)

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

[Ps 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

[Ps 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Ps Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Ps Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Ps Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Ps 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: `'[^\|>:<:\^*" \+,;=\[\] \.]{1,8}(\. [^\|>:<:\^*" \+,;=\[\] \.]{0,3})?|(!\$))\(\loc\.[_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Ps Schema](#)

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

[Ps Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Ps Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: `'[^\|><:/"+,;=\\[\.]{1,16}(\. [^\|><:/"+,;=\\[\.]{0,6})?(!\$)\(loc\[A-Za-z][0-9A-Za-z_]*\)'`.

See Also

[Ps 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

[Ps Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Ps Schema](#)

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](#)

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	Type	Description	Required
Id	String		Yes
ConfirmOverwrite	YesNoType		
ContinueOnError	YesNoType		
CreateOnInstall	YesNoType		
CreateOnReinstall	YesNoType	Specifies whether to create the database when the associated component is reinstalled. Setting CreateOnInstall to yes does not imply CreateOnReinstall is	

set to yes.
 CreateOnReinstall
 must be set in
 addition to
 CreateOnInstall for it
 to be created during
 both install and
 reinstall.

CreateOnUninstall	YesNoType		
Database	String	The name of the database. The value can be a literal value or derived from a Property element using the Formatted syntax.	Yes
DropOnInstall	YesNoType		
DropOnReinstall	YesNoType	Specifies whether to drop the database when the associated component is reinstalled. Setting DropOnInstall to yes does not imply DropOnReinstall is set to yes. DropOnReinstall must be set in addition to DropOnInstall for it to be dropped during both install and reinstall.	
DropOnUninstall	YesNoType		
Instance	String		
Server	String		Yes
User	String		

Remarks

Nesting SqlDatabase under a Component element will result in a SqlDatabase being installed to the machine as the package is installed.

Nesting SqlDatabase under Product, Fragment, or Module results in a database "locator" record being created in the SqlDatabase table. This means that the database itself is neither installed nor uninstalled by the MSI package. It does make the database available for referencing from a SqlString or SqlScript record. This allows MSI to install SqlScripts or SqlStrings to already existing databases on the machine. The install will fail if the database does not exist in these cases.

The User attribute references credentials specified in a User element. If a user is not specified then Windows Authentication will be used by default using the credentials of the user performing the install to execute sql strings, etc.

See Also

[Sql Schema](#), [User](#)

Version 3.0.5419.0

SqlFileSpec Element (Sql Extension)

Description

File specification for a Sql database.

Windows Installer references

None

Parents

[SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	ID of the file specification.	Yes
Filename	String	Specifies the operating-system file name for the database file.	Yes
GrowthSize	String	Specifies the growth increment of the database file. The 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 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](#)

SqlLogFileSpec Element (Sql Extension)

Description

File specification for a Sql database.

Windows Installer references

None

Parents

[SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Filename	String	Specifies the operating-system file name for the log file.	
GrowthSize	String	Specifies the growth increment of the log file. The 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](#)

SqlScript Element (Sql Extension)

Description

SQL Script

Windows Installer references

None

Parents

[Component](#), [SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	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	YesNoType	Specifies whether to execute the script when the associated component is reinstalled. Setting ExecuteOnInstall to yes does not imply ExecuteOnReinstall is set to yes. ExecuteOnReinstall must be set in addition to ExecuteOnInstall for it to be executed during both install and reinstall. This attribute is mutually exclusive with the RollbackOnInstall, RollbackOnReinstall and RollbackOnUninstall attributes.
ExecuteOnUninstall	YesNoType	Specifies to execute the script 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 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	Specifies the order to run the SQL Scripts. It is recommended that rollback scripts be scheduled before their complementary execution script. This order is also relative across the SqlString element.
SqlDb	String	required when not child of SqlDatabase
User	String	

See Also
[Sql Schema](#)

SqlString Element (Sql Extension)

Description

SQL String

Windows Installer references

None

Parents

[Component](#), [SqlDatabase](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String		Yes
ContinueOnError	YesNoType	Continue executing strings even if this one fails.	
ExecuteOnInstall	YesNoType	Specifies to execute the string when the associated component is installed. This attribute is mutually exclusive with the RollbackOnInstall, RollbackOnReinstall and RollbackOnUninstall attributes.	
ExecuteOnReinstall	YesNoType	Specifies whether to execute the string	

when the associated component is reinstalled. Setting ExecuteOnInstall to yes does **not** imply ExecuteOnReinstall is set to yes. ExecuteOnReinstall must be set in addition to ExecuteOnInstall for it to be executed during both install and reinstall. This attribute is mutually exclusive with the RollbackOnInstall, RollbackOnReinstall and RollbackOnUninstall attributes.

ExecuteOnUninstall	YesNoType	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 Specifies the order

to run the SQL Strings. It is recommended that rollback strings be scheduled before their complementary execution string. This order is also relative across the SqlScript element.

SQL	String	Yes
SqlDb	String	
User	String	

See Also
[Sql Schema](#)

Version 3.0.5419.0

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Sql 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

[Sql Schema](#)

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

[Sql 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

[Sql 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Sql Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Sql Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Sql Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Sql 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: `'[^\\?|><:\^*" \+,;=\[\] \.]{1,8}(\. [^\\?|><:\^*" \+,;=\[\] \.]{0,3})?(![!$])\(\loc\.[_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Sql 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

[Sql Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Sql Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: '[^\\|><:"'+,;=\\[\.]{1,16}\\. [^\\|><:"'+,;=\\[\.]{0,6}?|(!\$)\\(loc\\. [A-Za-z][0-9A-Za-z_]*\\)'

See Also

[Sql 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

[Sql Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Sql Schema](#)

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](#)
- [EventManifest](#)
- [EventSource](#)
- [FileShare](#)
- [FileSharePermission](#)
- [Group](#)
- [GroupRef](#)
- [InternetShortcut](#)
- [PerfCounter](#)
- [PerfCounterManifest](#)
- [PerformanceCategory](#)
- [PerformanceCounter](#)
- [PermissionEx](#)
- [ServiceConfig](#)
- [User](#)
- [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	Type	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 deferred 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 executable to be closed. This should only be file name.	Yes

See Also
[Util Schema](#)

EventManifest Element (Util Extension)

Description

Used to install Event Manifests.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	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](#)

Version 3.0.5419.0

EventSource Element (Util Extension)

Description

Creates an event source.

Windows Installer references

None

Parents

[Component](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
CategoryCount	Integer	The number of categories in CategoryMessageFile. CategoryMessageFile must be specified too.
CategoryMessageFile	String	Name of the category message file. CategoryCount must be specified too. Note that this is a formatted field, so you can use [#fileId] syntax to refer to a file being installed also written as a REG_EXPAND_SZ string, so you can use %environment_variable% syntax to refer to a file already present on the user's machine.
EventMessageFile	String	Name of the event message file. Note that this is a formatted field, so you can use [#fileId] syntax to refer to a file being installed. It is also

written as a REG_EXPAND_S string, so you can use %environment_variable% syntax to refer to a file already present on the user's machine.

KeyPath	YesNoType	Marks the EventSource register the key path of the component belongs to.
Log	String	Name of the event source's log.
Name	String	Name of the event source.
ParameterMessageFile	String	Name of the parameter message file. Note that this is a formatted field, so you can use [#fileId] syntax to refer to a file being installed also written as a REG_EXPAND_S string, so you can use %environment_variable% syntax to refer to a file already present on the user's machine.
SupportsErrors	YesNoType	Equivalent to EVENTLOG_ERROR_TYPE.
SupportsFailureAudits	YesNoType	Equivalent to EVENTLOG_AUDIT_FAILURE.
SupportsInformationals	YesNoType	Equivalent to EVENTLOG_INFORMATIONAL.
SupportsSuccessAudits	YesNoType	Equivalent to EVENTLOG_AUDIT_SUCCESS.
SupportsWarnings	YesNoType	Equivalent to EVENTLOG_WARNING_TYPE.

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

See Also

[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

Attributes

Name	Type	Description
ChangePermission	YesNoType	
CreateChild	YesNoType	For a directory, the right to create a subdirectory. Only valid under a 'CreateFolder' parent.
CreateFile	YesNoType	For a directory, the right to create a file in the directory. Only valid under a 'CreateFolder' parent.
Delete	YesNoType	
DeleteChild	YesNoType	For a directory, the right to delete the directory and all the files it contains, including read-only files. Only valid under a 'CreateFolder' parent.
GenericAll	YesNoType	

GenericExecute	YesNoType	
GenericRead	YesNoType	specifying this will fail to grant access
GenericWrite	YesNoType	
Read	YesNoType	
ReadAttributes	YesNoType	
ReadExtendedAttributes	YesNoType	
ReadPermission	YesNoType	
Synchronize	YesNoType	
TakeOwnership	YesNoType	
Traverse	YesNoType	For a directory, the right to tra the directory. By default, user: assigned the BYPASS_TRAVERSE_CHEC privilege, which ignores the FILE_TRAVERSE access righ Only valid under a 'CreateFol parent.
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	Type	Description	Required
Id	String	Unique identifier in your installation package for this group.	Yes
Domain	String	An optional Formatted string that specifies the domain for the group.	
Name	String	A Formatted 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

[User](#)

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

Attributes

Name	Type	Description	Required
Id	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 when the user selects the shortcut. Windows opens the	Yes

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: <i>url</i> Creates .url files using IUniformResourceLocatorW. <i>link</i> Creates .lnk 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 [PerformanceCounter](#) element instead.

Windows Installer references

None

Parents

[File](#)

Inner Text

None

Children

None

Attributes

Name	Type	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

Attributes

Name	Type	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 registration 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](#)

Version 3.0.5419.0

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)

Attributes

Name	Type	Description
Close	String	Function ent the Library E when closing performance default is "ClosePerfor which should all managed performance
Collect	String	Function ent the Library E when collect the performe The default i

		"CollectPerf which should all managed performance
DefaultLanguage	PerformanceCounterLanguageType	Default lang performance and containe names and l
Help	String	Optional hel performance category.
Id	String	Unique iden installation p this perform; category.
Library	String	DLL that cor performance default is "ne which should all managed performance
MultInstance	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 L when openir performance

default is
"OpenPerfor
which shoul
all managed
performance

See Also
[Util Schema](#)

Version 3.0.5419.0

PerformanceCounter Element (Util Extension)

Description

Creates a performance counter in a performance category.

Windows Installer references

None

Parents

[PerformanceCategory](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description
Help	String	Optional help text for the performance counter.
Language	PerformanceCounterLanguageType	Language for the performance counter name and help. The default is to use the parent PerformanceCategory element's DefaultLanguage attribute.
Name	String	Name for the performance counter.
Type	PerformanceCounterTypesType	Type of the performance counter.

See Also
[Util Schema](#)

Version 3.0.5419.0

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

Attributes

Name	Type	Description
Append	YesNoType	
ChangePermission	YesNoType	
CreateChild	YesNoType	For a directory, the right subdirectory. Only valid 'CreateFolder' parent.
CreateFile	YesNoType	For a directory, the right file in the directory. Only under a 'CreateFolder'
CreateLink	YesNoType	
CreateSubkeys	YesNoType	

Delete	YesNoType	
DeleteChild	YesNoType	For a directory, the right directory and all the file contains, including recursive. Only valid under a 'Create' parent.
Domain	String	
EnumerateSubkeys	YesNoType	
Execute	YesNoType	
GenericAll	YesNoType	
GenericExecute	YesNoType	
GenericRead	YesNoType	specifying this will fail to access
GenericWrite	YesNoType	
Notify	YesNoType	
Read	YesNoType	
ReadAttributes	YesNoType	
ReadExtendedAttributes	YesNoType	
ReadPermission	YesNoType	
ServiceChangeConfig	YesNoType	Required to call the ChangeServiceConfig or ChangeServiceConfig2 to change the service configuration. Only valid under a 'Service' parent.
ServiceEnumerateDependents	YesNoType	Required to call the EnumDependentServices to enumerate all the services dependent on the service. Only valid under a 'Service' parent.
ServiceInterrogate	YesNoType	Required to call the ControlServiceStatus function to ask the service its status immediately. Only valid under a 'ServiceInstall' parent.

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 fi query the service config Only valid under a 'Ser parent.
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	YesNoType	Required to call the Co function to stop the ser valid under a 'ServiceIn parent.
ServiceUserDefinedControl	YesNoType	Required to call the Co function to specify a us control code. Only valid 'ServiceInstall' parent.
Synchronize	YesNoType	
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 'Cre parent.

User	String
Write	YesNoType
WriteAttributes	YesNoType
WriteExtendedAttributes	YesNoType

See Also

[Util Schema](#)

Version 3.0.5419.0

ServiceConfig Element (Util Extension)

Description

Service configuration information for failure actions.

Windows Installer references

None

Parents

[Component](#), [ServiceInstall](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
FirstFailureActionType	Enumeration	Action to take on the first failure of the service. This attribute's value must be one of the following: <i>none</i> <i>reboot</i> <i>restart</i> <i>runCommand</i>	Yes
ProgramCommandLine	String	If any of the three *ActionType attributes is "runCommand",	

		this specifies the command to run when doing so.	
RebootMessage	String	If any of the three *ActionType attributes is "reboot", this specifies the message to broadcast to server users before doing so.	
ResetPeriodInDays	Integer	Number of days after which to reset the failure count to zero if there are no failures.	
RestartServiceDelayInSeconds	Integer	If any of the three *ActionType attributes is "restart", this specifies the number of seconds to wait before doing so.	
SecondFailureActionType	Enumeration	Action to take on the second failure of the service. This attribute's value must be one of	Yes

		the following: <i>none</i> <i>reboot</i> <i>restart</i> <i>runCommand</i>	
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: <i>none</i> <i>reboot</i> <i>restart</i> <i>runCommand</i>	Yes

Remarks

Nesting a ServiceConfig element under a ServiceInstall element will result in the service being installed to be configured.

Nesting a ServiceConfig element under a component element will result in an already installed service to be configured. If the service does not exist prior to the install of the MSI package, the install will fail.

See Also

[Util Schema](#)

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)

Attributes

Name	Type	Description
Id	String	
CanNotChangePassword	YesNoType	The user cannot change the account's password. Equivalent to UF_PASSWD_CANT_CHAN
CreateUser	YesNoType	Indicates whether or not to create the user. User creation can be skipped if all that is desired is to join a user to groups.
Disabled	YesNoType	The account is disabled. Equivalent to

		UF_ACCOUNTDISABLE.
Domain	String	A Formatted string that contains the local machine or Active Directory domain for the user.
FailIfExists	YesNoType	Indicates if the install should fail if the user already exists.
LogonAsService	YesNoType	Indicates whether or not the user can logon as a service. User creation can be skipped if all that is desired is to set this access right on the user.
Name	String	A Formatted string that contains the name of the user account.
Password	String	Usually a Property that is passed in on the command-line to keep it more secure.
PasswordExpired	YesNoType	Indicates whether the user must change their password on their first login.
PasswordNeverExpires	YesNoType	The account's password never expires. Equivalent to UF_DONT_EXPIRE_PASSWORD.
RemoveOnUninstall	YesNoType	Indicates whether the user account should be removed or left behind on uninstall.
UpdateIfExists	YesNoType	Indicates if the user account properties should be updated if 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)

Attributes

Name	Type	Description	Required
Id	String	Identifier for xml file modification.	Yes
Action	Enumeration	This attribute's value must be one of the following: <i>create</i> <i>delete</i>	
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: <i>element</i> <i>value</i> <i>document</i>	
On	Enumeration	This attribute's	

value must be one of the following:
install
uninstall

PreserveModifiedDate	YesNoType	Specifies whether 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](#)

Version 3.0.5419.0

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

Attributes

Name	Type	Description	Required
Id	String	Identifier for xml file modification.	Yes
Action	Enumeration	The type of modification to be made to the XML file when the component is installed. This attribute's value must be one of the following: <i>createElement</i> Creates a new element under the element specified in	Yes

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: <i>XPath</i>

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](#)

Version 3.0.5419.0

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]{4}\-?[0-9A-Fa-f]{12}[)]?|[(]?[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

[Util 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

[Util Schema](#)

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

[Util 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

[Util 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]*|(![\$])\((?:loc|bind)\.[_A-Za-z][0-9A-Za-z_\.]+\)'.

See Also

[Util Schema](#)

LongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Util Schema](#)

PatchClassificationType (Simple Type)

Description

Category of update.

Enumeration Type

Possible values: {Critical Update, Hotfix, Security Rollup, Service Pack, Update, Update Rollup}

See Also

[Util Schema](#)

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

[Util Schema](#)

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, counterMultiTimer, counterMultiTimer100Ns, counterMultiTimerInverse, counterMultiTimer100NsInverse, elapsedTime, sampleBase, rawBase, counterMultiBase, rateOfCountsPerSecond64, rateOfCountsPerSecond32, countPerTimeInterval64, countPerTimeInterval32, sampleCounter, counterDelta64, numberOfItems64, numberOfItems32, numberOfItemsHEX64, numberOfItemsHEX32}

See Also

[Util Schema](#)

RegistryRootType (Simple Type)

Description

Values of this type represent possible registry roots.

Enumeration Type

Possible values: {HKMU, HKCR, HKCU, HKLM, HKU}

See Also

[Util 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: `'[^\|>:<:*"\+,\;=\[\]\.]{1,8}(\. [^\|>:<:*"\+,\;=\[\]\.]{0,3})?(!\$)\(loc\[_A-Za-z][0-9A-Za-z_\.]*\)'`.

See Also

[Util 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

[Util Schema](#)

WildcardLongFileNameType (Simple Type)

Description

Values of this type will look like: "Long File Name.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

[Util Schema](#)

WildcardShortFileNameType (Simple Type)

Description

Values of this type will look like: "File?.*". 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: `'[^\\|><:/"+,;=\\[\.]{1,16}(\. [^\\|><:/"+,;=\\[\.]{0,6})?(!\$)\\(loc\\. [A-Za-z][0-9A-Za-z_]*\\)'`

See Also

[Util 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

[Util Schema](#)

YesNoType (Simple Type)

Description

Values of this type will either be "yes" or "no".

Enumeration Type

Possible values: {no, yes}

See Also

[Util Schema](#)

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](#)
- [HelpFile](#)
- [HelpFileRef](#)
- [HelpFilter](#)
- [HelpFilterRef](#)
- [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	Type	Description	Required
Id	String	Primary Key for HelpNamespace.	Yes
Description	String	Friendly name for Namespace.	
Name	String	Internal Microsoft Help ID for this Namespace.	Yes
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](#)

Version 3.0.5419.0

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'](#)

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](#)

Version 3.0.5419.0

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	Type	Description	Required
Id	String	Primary Key for HelpFile Table.	Yes
Any attribute namespace='###other' processContents='lax'			

See Also

[Vs Schema](#)

Version 3.0.5419.0

HelpFilter Element (Vs Extension)

Description

Filter for Help Namespace.

Windows Installer references

None

Parents

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

Inner Text

None

Children

None

Attributes

Name	Type	Description	Required
Id	String	Primary Key for HelpFilter.	Yes
FilterDefinition	String	Query String for Help Filter.	
Name	String	Friendly name for Filter.	Yes
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](#)

Version 3.0.5419.0

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	Type	Description	Required
Id	String	Primary Key for HelpFilter.	Yes
Any attribute namespace='###other' processContents='lax'			

See Also

[Vs Schema](#)

Version 3.0.5419.0

PlugCollectionInto Element (Vs Extension)

Description

Plugin for Help Namespace.

Windows Installer references

None

Parents

[HelpCollection](#)

Inner Text

None

Children

None

Attributes

Name	Type	Description	Req
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	Foreign 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](#)

Advanced WiX Topics

This section covers the following advanced WiX topics:

[Specifying Cultures to Build](#)

[Specifying Source Files](#)

[Optimizing Builds](#)

[Adding Custom Actions](#)

[Standard Custom Actions](#)

[WixUI Dialog Library](#)

[Extensions](#)

[Patch Building](#)

[Code Pages](#)

Specifying Cultures to Build

Specifying Cultures to build on the Command Line

You can specify a specific culture for light.exe to build using the culture switch:

```
light.exe myinstaller.wixobj -cultures:en-us -ext WixUIExtension  
-out myinstaller-en-us.msi
```

This will cause light to build an en-us installer using the en-us resources from WixUIExtension.

You can still use cultures when specifying localization files:

```
light.exe myinstaller.wixobj -cultures:en-us -loc mystrings_en-US.w  
-loc mystrings_fr-FR.wxl -out myinstaller-en-us.msi
```

This will cause light to build an en-us installer using the en-us resources from the specified en-US .wxi file. Note that when specifying -cultures any wxl files specified with the -loc switch that do not map will be ignored (mystrings_fr-FR.wxl in this case.)

The neutral (invariant) culture can be specified by using *neutral*:

```
light.exe myinstaller.wixobj -cultures:neutral -loc mystrings_en-US  
-loc mystrings_fr-FR.wxl -loc mystrings.wxl -out myinstalle
```

This will cause light to build a neutral installer using the neutral resources from the mystrings.wxl file.

You can use cultures and localization files together to specify fallback cultures:

```
light.exe myinstaller.wixobj -cultures:en-us;en -loc mystrings_en-U  
-loc mystrings_en.wxl -ext WixUIExtension -out myinstaller-
```

This will cause light to build an en-us installer first using localization variables from the en-US localization file (mystrings_en-US.wxl), then the en localization file (mystrings_en.wxl), then finally WixUIExtension.

Specifying Cultures to build in Visual Studio

During the development of your installer you may want to temporarily disable building some of the languages to speed up build time. You can do this by going to **Project > Projectname Properties** on the menu and selecting the **Build** tab. In the **Cultures to build** field enter the semicolon list of cultures or culture groups you would like built.

Cultures to build may be used to specify cultures to build when a .wxi file is not provided for a target culture. For example, to build an en-US installer and an ru-RU installer when only an ru-RU .wxi file is provided, specify en-US;ru-RU. Wix localization variables for the ru-RU installer will first come from the provided .wxi file, then referenced WiX extensions (IE: WixUIExtension). Wix localization variables for the en-US installer will only come from referenced extensions.

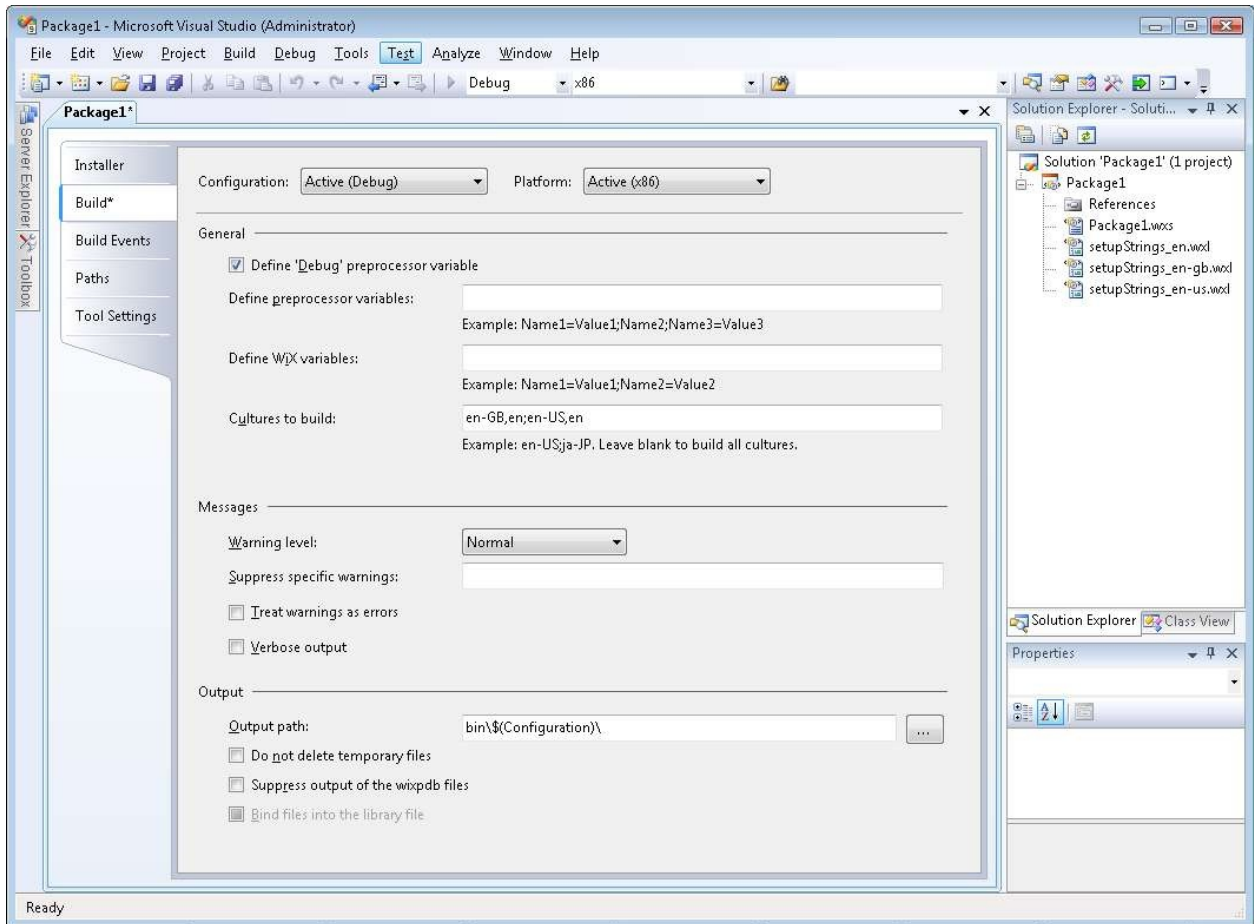
The neutral (invariant) culture can be specified by using *neutral*. To build English (United States), French (France), and neutral installers specify the following:

```
en-US;fr-FR;neutral
```

Cultures to build may also be used to specify how to use multiple WixL files to build a single installer. Each culture or culture group will build an individual MSI. A **culture group** consists of a list of cultures separated by *commas* and is useful for specifying fallback cultures used to locate WiX localization variables. Multiple culture groups may be separated by *semicolons* to build multiple outputs.

```
primary1, fallback1;primary2, fallback2
```

The example below demonstrates the settings needed to build two installers from three .wxi files. Both en-US and en-GB installers will be built, using three localization files: setupStrings_en-US.wxi, setupStrings_en-GB.wxi, and setupStrings_en.wxi. The sample uses two culture groups to share the neutral English translations between both of the fully localized installers.



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 [FileSource](#) attribute for the [Directory](#) and [DirectoryRef](#) 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 [FileSource](#) 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.

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.

Adding Custom Actions

Now that you're comfortable with the basics for creating Windows Installer packages, let's take it to the next level and add a CustomAction. This example will show how to author a binary CustomAction called "FooAction". A common sample people use is a dll customaction 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".

Rather than put the CustomAction definition in the same source file as our product definition, let's exercise a little modularity and create a new source file to define the CustomActions called "ca.wxs".

```
<?xml version='1.0'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
  <Fragment>
    <CustomAction Id='FooAction' BinaryKey='FooBinary' DllEntry='
      Return='check' />

    <Binary Id='FooBinary' SourceFile='foo.dll' />
  </Fragment>
</Wix>
```

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. 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'
    Version='1.0.0.0' Manufacturer='Microsoft Corporation'>
    <Package Description='My first Windows Installer package'
      Comments='This is my first attempt at creating a Window
        Manufacturer='Microsoft Corporation' InstallerVersion='

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

    <Directory Id='TARGETDIR' Name='SourceDir'>
      <Directory Id='ProgramFilesFolder' Name='PFiles'>
        <Directory Id='MyDir' Name='Test Program'>
```

```

        <Component Id='MyComponent' Guid='PUT-GUID-HERE'>
            <File Id='readme' Name='readme.txt' DiskId='1' So
        </Component>

        <Merge Id='MyModule' Language='1033' SourceFile='mod
    </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.

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 [Using Standard Custom Actions](#) topic.

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

[FileShare custom action](#) (located in WixUtilExtension) - create and configure file shares.

[Internet shortcut custom action](#) (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.

[Performance Counter custom action](#) (located in WixUtilExtension) - install and uninstall performance counters.

[Quiet Execution custom action](#) (located in WixUtilExtension) - launch console executables without displaying a window.

[Secure Objects custom action](#) (located in WixUtilExtension) - secure (using ACLs) objects that the [LockPermissions table](#) cannot.

[Service Configuration custom action](#) (located in WixUtilExtension) - configure attributes of a Windows service that the [ServiceInstall table](#) cannot.

[ShellExecute custom action](#) (located in WixUtilExtension) - launch document or URL targets via the Windows shell.

[User custom actions](#) (located in WixUtilExtension) - create and configure new users.

[WixDirectXExtension](#) - custom action that can be used to check the DirectX capabilities of the video card on the system.

[WixExitEarlyWithSuccess](#) (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.

[WixFailWhenDeferred](#) (located in WixUtilExtension) - custom action that can be used to simulate installation failures to test rollback scenarios.

[WixFirewallExtension](#) - Firewall custom action that can be used to add exceptions to the Windows Firewall.

[WixGamingExtension](#) - Gaming custom action that can be used to add icons and tasks to Windows Game Explorer.

[WixIISExtension](#) - Internet Information Services (IIS) custom actions that can be used to create and configure web sites, virtual directories, web applications, etc.

[WixNetFxExtension](#) - custom action to generate native code for .NET assemblies; properties to detect .NET Framework install state and service pack levels.

[WixSqlExtension](#) - SQL Server custom actions that can be used to create databases and execute SQL scripts and statements.

[WixVSEExtension](#) - 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.

```
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi' xmlns:util='h
  <Product Id='PutGuidHere' Name='TestUserProduct' Language='1033
    <Package Id='PUT-GUID-HERE' Description='Test User Package'
      <Directory Id='TARGETDIR' Name='SourceDir'>
        <Component Id='TestUserProductComponent' Guid='PutG
          <util:User Id='TEST_USER1' Name='testName1' Pas
        </Component>
      </Directory>

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

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 [<PropertyRef>](#) to the property you want to use and then include WixUtilExtensions when linking. For example:

```
<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_SERVER2	Equivalent to the GetSys SM_SERVER2 flag.
WIX_SUITE_SINGLEUSERS	Equivalent to the OSVEF VER_SUITE_SINGLEUSERS
WIX_SUITE_SMALLBUSINESS	Equivalent to the OSVEF VER_SUITE_SMALLBUSINESS
WIX_SUITE_SMALLBUSINESS_RESTRICTED	Equivalent to the OSVEF VER_SUITE_SMALLBUSINESS_RESTRICTED flag.
WIX_SUITE_STARTER	Equivalent to the GetSys SM_STARTER flag.
WIX_SUITE_STORAGE_SERVER	Equivalent to the OSVEF VER_SUITE_STORAGE_SERVER
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_SERVER

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. [This MSDN documentation](#) is a good place to start. See below for samples re-purposed from MSDN.

The RegisterPerfmon custom action - You can link with the WiXUtilExtension.dll to ensure that the custom actions are included in your final MSI. See [Using Standard Custom Actions](#) . 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\CurrentCon
        <Registry Id="Shared_r2" Root="HKLM" Key="SYSTEM\CurrentCon
        <Registry Id="Shared_r3" Root="HKLM" Key="SYSTEM\CurrentCon
        <Registry Id="Shared_r4" Root="HKLM" Key="SYSTEM\CurrentCon

      <File Id="PERFDLL.DLL" Name="MyPerfDll.dll" Source="x86\debu

      <File Id="PERFCOUNTERS.H" Name="PerfCounters.h" Source="x86\
      <File Id="PERFCOUNTERS.INI" Name="PerfCounters.ini" Source="
        <PerfCounter Name="MyApplication" />
      </File>

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

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

[languages]
009=English
004=Chinese

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

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

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

Quiet Execution 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:

```
<Property Id="QtExecCmdLine" Value="command line to run"/>
<CustomAction Id="QtExecExample" BinaryKey="WixCA" DllEntry="CAQuie
.
.
.
<InstallExecuteSequence>
  <Custom Action="QtExecExample" After="TheActionYouWantItAfter"/>
</InstallExecuteSequence>
```

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:

```
<Property Id="QtExecDeferredExample" Value="command line to run"/>
<CustomAction Id="QtExecDeferredExample" BinaryKey="WixCA" DllEntry
    Execute="deferred" Return="check" Impersonate="no"/>
.
.
.
<InstallExecuteSequence>
    <Custom Action="QtExecDeferredExample" After="TheActionYouWantI
</InstallExecuteSequence>
```

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:

```
<CustomAction Id="QtExecDeferredExampleWithProperty_Cmd" Property="
    Value="&quot;[#MyExecutable.exe]&quot;" Execute="imme
<CustomAction Id="QtExecDeferredExampleWithProperty" BinaryKey="Wix
    Execute="deferred" Return="check" Impersonate="no"/>
.
.
.
<InstallExecuteSequence>
    <Custom Action="QtExecDeferredExampleWithProperty_Cmd" After="C
    <Custom Action="QtExecDeferredExampleWithProperty" After="TheAc
</InstallExecuteSequence>
```

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:

```
<Property Id="QtExec64CmdLine" Value="64-bit command line to run"/>
<CustomAction Id="QtExecExample" BinaryKey="WixCA" DllEntry="CAQuietExec64"
.
.
.
<CustomAction Id="QtExecDeferredExampleWithProperty_Cmd" Property="QtExec64CmdLine"
Value="&quot;[#MyExecutable.exe]&quot;" Execute="immediate"
<CustomAction Id="QtExecDeferredExampleWithProperty" BinaryKey="WixCA"
Execute="deferred" Return="check" Impersonate="no"/>
.
.
.
<InstallExecuteSequence>
  <Custom Action="QtExecExample" After="TheImmediateActionYouWant"
.
  <Custom Action="QtExecDeferredExampleWithProperty_Cmd" After="TheDeferredExampleWithProperty_Cmd"
  <Custom Action="QtExecDeferredExampleWithProperty" After="TheDeferredExampleWithProperty_Cmd"
</InstallExecuteSequence>
```

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 [How To: Run the Installed Application After Setup](#) 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 capabilities as <i>major*100 + minor</i> . For example, a system with a pixel shader model 3.0-compliant video card has a WIX_DIRECTX_PIXELSHADERVERSION value of 300.
--------------------------------	---

WIX_DIRECTX_VERTEXSHADERVERSION	Vertex shader version capabilities as <i>major*100 + minor</i> . For example, a system with a vertex shader model 3.0-compliant video card has a WIX_DIRECTX_VERTEXSHADERVERSION 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 Wix 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:

```
<PropertyRef Id="WIX_DIRECTX_PIXELSHADERVERSION" />
```

```
<CustomAction Id="CA_CheckPixelShaderVersion" Error="[ProductName]
<InstallExecuteSequence>
  <Custom Action="CA_CheckPixelShaderVersion" After="WixQueryDirect
    <![CDATA[WIX_DIRECTX_PIXELSHADERVERSION < 300]]>
  </Custom>
</InstallExecuteSequence>

<InstallUISequence>
  <Custom Action="CA_CheckPixelShaderVersion" After="WixQueryDirect
    <![CDATA[WIX_DIRECTX_PIXELSHADERVERSION < 300]]>
  </Custom>
</InstallUISequence>
```

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"  
</Upgrade>
```

WixFailWhenDeferred Custom Action

When authoring [deferred custom actions](#) (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
6. 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 [InstallFinalize](#) 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:

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

WixGamingExtension

The [WixGamingExtension](#) 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 [Getting Started With Game Explorer](#). 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 [The Game-Definition-File \(GDF\) Schema](#) and [GDF Delivery and Localization](#). 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="ye
    <gaming:Game Id="985D5FD3-FC40-4CE9-9EE5-F2AAAB959230">
        ...
    </File>
```

The Game/@Id attribute is used as the InstanceID attribute discussed [here](#), rather than generating new GUIDs at install time, which would require persisting the generated GUID and loading it for uninstall and maintenance mode.

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 [Supporting an Upgrade from Windows XP to Windows Vista](#) 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 [Game Explorer Tasks](#). In WixGameExtension, PlayTask and SupportTask are child elements of the Game element:

```
<File Id="MyGameExeFile" Name="passenger_simulator.exe" KeyPath="ye
  <gaming:Game Id="985D5FD3-FC40-4CE9-9EE5-F2AAAB959230">
    <gaming:PlayTask Name="Play" Arguments="-go" />
    <gaming:SupportTask Name="Help!" Address="http://example.co
      ...
    ...
  </File>
```

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 [Shortcuts](#) and SupportTask into WixUtilExtension [InternetShortcuts](#). 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 [Rich Saved Games](#). If your game supports rich saved games, you can register it for the rich saved-game preview using the WixGamingExtension IsRichSavedGame attribute on the [Extension element](#):

```
<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 [WixNetfxExtension](#) includes a set of custom actions to compile native images using Ngen.exe. For an example, see [How To: NGen managed assemblies during installation](#).

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 directory for all versions of the .NET Framework (%windir%\Microsoft.NET\Framework

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

Property name	Meaning
NETFRAMEWORK10	Set to 3321-3705 if the .NET Framework 1.0 is installed (not set otherwise).
NETFRAMEWORK10INSTALLROOTDIR	Set to the installation directory for .NET Framework 1.0 (%windir%\Microsoft.NET\Fram

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 Framework 1.1 is installed (not set otherwise).
NETFRAMEWORK11_SP_LEVEL	Indicates the service pack level for .NET Framework 1.1.

NETFRAMEWORK11INSTALLROOTDIR	Set to the installation directory for the .NET Framework 1.1 (%windir%\Microsoft.NET\Fi
NETFRAMEWORK11_ZH_CN_LANGPACK	Set to #1 if the .NET Framework (Simplified) language pack is installed (0 otherwise).
NETFRAMEWORK11_ZH_TW_LANGPACK	Set to #1 if the .NET Framework (Traditional) language pack is installed (0 otherwise).
NETFRAMEWORK11_CS_CZ_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_DA_DK_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_NL_NL_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_FI_FI_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_FR_FR_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_DE_DE_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_EL_GR_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).
NETFRAMEWORK11_HU_HU_LANGPACK	Set to #1 if the .NET Framework language pack is installed (0 otherwise).

NETFRAMEWORK11_IT_IT_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_JA_JP_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_KO_KR_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_NB_NO_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_PL_PL_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_PT_BR_LANGPACK	Set to #1 if the .NET Framework (Brazil) language pack is installed (otherwise).
NETFRAMEWORK11_PT_PT_LANGPACK	Set to #1 if the .NET Framework (Portugal) language pack is installed (otherwise).
NETFRAMEWORK11_RU_RU_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_ES_ES_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_SV_SE_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK11_TR_TR_LANGPACK	Set to #1 if the .NET Framework 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 Framework is installed (not set otherwise).
NETFRAMEWORK20_SP_LEVEL	Indicates the service pack level of the .NET Framework 2.0.
NETFRAMEWORK20INSTALLROOTDIR	Set to the installation directory of the .NET Framework 2.0 (%windir%\Microsoft.NET\Framework64\).NET\Framework64\
NETFRAMEWORK20INSTALLROOTDIR64	Set to the installation directory of the .NET Framework 2.0 (%windir%\Microsoft.NET\Framework\).NET\Framework\
NETFRAMEWORK20_ZH_CN_LANGPACK	Set to #1 if the .NET Framework (Simplified) language pack is installed (not set otherwise).
NETFRAMEWORK20_ZH_TW_LANGPACK	Set to #1 if the .NET Framework (Traditional) language pack is installed (not set otherwise).
NETFRAMEWORK20_CS_CZ_LANGPACK	Set to #1 if the .NET Framework (Czech) language pack is installed (not set otherwise).
NETFRAMEWORK20_DA_DK_LANGPACK	Set to #1 if the .NET Framework (Danish) language pack is installed (not set otherwise).
NETFRAMEWORK20_NL_NL_LANGPACK	Set to #1 if the .NET Framework (Dutch) language pack is installed (not set otherwise).

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

	(Portugal) language pack is otherwise).
NETFRAMEWORK20_RU_RU_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK20_ES_ES_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK20_SV_SE_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK20_TR_TR_LANGPACK	Set to #1 if the .NET Framework 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 Framework installed (not set otherwise).
NETFRAMEWORK30_SP_LEVEL	Indicates the service pack level Framework 3.0. This value v service pack is installed.
NETFRAMEWORK30INSTALLROOTDIR	Set to the installation directory Framework 3.0 (%windir%\Microsoft.NET\F
NETFRAMEWORK30INSTALLROOTDIR64	Set to the installation directory .NET Framework 3.0 (%windir%\Microsoft.NET\F

NETFRAMEWORK30_ZH_CN_LANGPACK	Set to #1 if the .NET Framework (Simplified) language pack is installed (otherwise).
NETFRAMEWORK30_ZH_TW_LANGPACK	Set to #1 if the .NET Framework (Traditional) language pack is installed (otherwise).
NETFRAMEWORK30_CS_CZ_LANGPACK	Set to #1 if the .NET Framework Czech language pack is installed (otherwise).
NETFRAMEWORK30_DA_DK_LANGPACK	Set to #1 if the .NET Framework Danish language pack is installed (otherwise).
NETFRAMEWORK30_NL_NL_LANGPACK	Set to #1 if the .NET Framework Dutch language pack is installed (otherwise).
NETFRAMEWORK30_FI_FI_LANGPACK	Set to #1 if the .NET Framework Finnish language pack is installed (otherwise).
NETFRAMEWORK30_FR_FR_LANGPACK	Set to #1 if the .NET Framework French language pack is installed (otherwise).
NETFRAMEWORK30_DE_DE_LANGPACK	Set to #1 if the .NET Framework German language pack is installed (otherwise).
NETFRAMEWORK30_EL_GR_LANGPACK	Set to #1 if the .NET Framework Greek language pack is installed (otherwise).
NETFRAMEWORK30_HU_HU_LANGPACK	Set to #1 if the .NET Framework Hungarian language pack is installed (otherwise).
NETFRAMEWORK30_IT_IT_LANGPACK	Set to #1 if the .NET Framework Italian language pack is installed (otherwise).

NETFRAMEWORK30_JA_JP_LANGPACK	Set to #1 if the .NET Framework Japanese language pack is otherwise).
NETFRAMEWORK30_KO_KR_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK30_NB_NO_LANGPACK	Set to #1 if the .NET Framework Norwegian language pack is otherwise).
NETFRAMEWORK30_PL_PL_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK30_PT_BR_LANGPACK	Set to #1 if the .NET Framework Portuguese (Brazil) language (not set otherwise).
NETFRAMEWORK30_PT_PT_LANGPACK	Set to #1 if the .NET Framework Portuguese (Portugal) language installed (not set otherwise).
NETFRAMEWORK30_RU_RU_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK30_ES_ES_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK30_SV_SE_LANGPACK	Set to #1 if the .NET Framework language pack is installed (r
NETFRAMEWORK30_TR_TR_LANGPACK	Set to #1 if the .NET Framework 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 Framework is installed (not set otherwise).
NETFRAMEWORK35_SP_LEVEL	Indicates the service pack level of the .NET Framework 3.5.
NETFRAMEWORK35INSTALLROOTDIR	Set to the installation directory of the .NET Framework 3.5 (%windir%\Microsoft.NET\Framework35\)
NETFRAMEWORK35INSTALLROOTDIR64	Set to the installation directory of the .NET Framework 3.5 (%windir%\Microsoft.NET\Framework64\)
NETFRAMEWORK35_ZH_CN_LANGPACK	Set to #1 if the .NET Framework (Simplified) language pack is installed (not set otherwise).
NETFRAMEWORK35_ZH_TW_LANGPACK	Set to #1 if the .NET Framework (Traditional) language pack is installed (not set otherwise).
NETFRAMEWORK35_CS_CZ_LANGPACK	Set to #1 if the .NET Framework (Czech) language pack is installed (not set otherwise).
NETFRAMEWORK35_DA_DK_LANGPACK	Set to #1 if the .NET Framework (Danish) language pack is installed (not set otherwise).
NETFRAMEWORK35_NL_NL_LANGPACK	Set to #1 if the .NET Framework (Dutch) language pack is installed (not set otherwise).

NETFRAMEWORK35_FI_FI_LANGPACK	Set to #1 if the .NET Framework Finnish language pack is installed (otherwise).
NETFRAMEWORK35_FR_FR_LANGPACK	Set to #1 if the .NET Framework French language pack is installed (otherwise).
NETFRAMEWORK35_DE_DE_LANGPACK	Set to #1 if the .NET Framework German language pack is installed (otherwise).
NETFRAMEWORK35_EL_GR_LANGPACK	Set to #1 if the .NET Framework Greek language pack is installed (otherwise).
NETFRAMEWORK35_HU_HU_LANGPACK	Set to #1 if the .NET Framework Hungarian language pack is installed (otherwise).
NETFRAMEWORK35_IT_IT_LANGPACK	Set to #1 if the .NET Framework Italian language pack is installed (otherwise).
NETFRAMEWORK35_JA_JP_LANGPACK	Set to #1 if the .NET Framework Japanese language pack is installed (otherwise).
NETFRAMEWORK35_KO_KR_LANGPACK	Set to #1 if the .NET Framework Korean language pack is installed (otherwise).
NETFRAMEWORK35_NB_NO_LANGPACK	Set to #1 if the .NET Framework Norwegian language pack is installed (otherwise).
NETFRAMEWORK35_PL_PL_LANGPACK	Set to #1 if the .NET Framework Polish language pack is installed (otherwise).
NETFRAMEWORK35_PT_BR_LANGPACK	Set to #1 if the .NET Framework Portuguese (Brazil) language pack is installed (otherwise).

NETFRAMEWORK35_PT_PT_LANGPACK	Set to #1 if the .NET Framework Portuguese (Portugal) language pack is installed (not set otherwise).
NETFRAMEWORK35_RU_RU_LANGPACK	Set to #1 if the .NET Framework Russian language pack is installed (not set otherwise).
NETFRAMEWORK35_ES_ES_LANGPACK	Set to #1 if the .NET Framework Spanish language pack is installed (not set otherwise).
NETFRAMEWORK35_SV_SE_LANGPACK	Set to #1 if the .NET Framework Swedish language pack is installed (not set otherwise).
NETFRAMEWORK35_TR_TR_LANGPACK	Set to #1 if the .NET Framework Turkish language pack is installed (not set otherwise).
NETFRAMEWORK35_CLIENT	Set to #1 if the .NET Framework 3.5 client profile is installed (not set otherwise).
NETFRAMEWORK35_CLIENT_SP_LEVEL	Indicates the service pack level of the .NET Framework 3.5 client profile.

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.

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 [WixVSExtension](#) 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
VS2003DEVENV	Full path to devenv.exe for Visual Studio .NET 2003 if it is installed on the system.
JSHARP_REDIST_11_INSTALLED	Indicates whether or not the J# redistributable package 1.1 is installed on the system.

Here is a complete list of properties for the **Visual Studio 2005** product family:

Property name	Meaning
VS2005DEVENV	Full path to devenv.exe for Visual Studio 2005 if it is installed on the system.
VS2005_ITEMTEMPLATES_DIR	Full path to the Visual Studio 2005 item templates directory.
VS2005_PROJECTTEMPLATES_DIR	Full path to the

	Visual Studio 2005 project templates directory.
VS2005_SCHEMAS_DIR	Full path to the Visual Studio 2005 XML schemas directory.
VS2005PROJECTAGGREGATOR2	Indicates whether or not the Visual Studio 2005 project aggregator for managed code add-ins is installed on the system.
VS2005_ROOT_FOLDER	Full path to the Visual Studio 2005 root installation directory.
VB2005EXPRESS_IDE	Full path to vbexpress.exe if Visual Basic 2005 Express Edition is installed on the system.
VS2005_IDE_VB_PROJECTSYSTEM_INSTALLED	Indicates

	whether Visual Studio 2005 Standard Edition or higher is installed and the Visual Basic project system is installed for
VC2005EXPRESS_IDE	Full path to vcexpress.exe if Visual C++ 2005 Express Edition is installed on this system.
VS2005_IDE_VC_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2005 Standard Edition or higher is installed and the Visual C++ project system is installed for
VCSHARP2005EXPRESS_IDE	Full path to vcsharpexpress.exe if Visual C# 2005 Express Edition is installed on this system.

VS2005_IDE_VCSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2005 Standard Edition or higher is installed and the Visual C# project system is installed for
VJSHARP2005EXPRESS_IDE	Full path to vjsexpress.exe if Visual J# 2005 Express Edition is installed on the system.
VS2005_IDE_VJSHARP_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2005 Standard Edition or higher is installed and the Visual J# project system is installed for
VWD2005EXPRESS_IDE	Full path to vwdexpress.exe if Visual Web Developer 2005 Express Edition is installed on the system.

VS2005_IDE_VWD_PROJECTSYSTEM_INSTALLED	Indicates whether Visual Studio 2005 Standard Edition or higher is installed and the Visual Web Developer project system is installed on the system.
VS2005_IDE_VSTS_TESTSYSTEM_INSTALLED	Indicates whether or not the Visual Studio Team Test project system is installed on the system.
VSEXTENSIONS_FOR_NETFX30_INSTALLED	Indicates whether or not the Visual Studio 2008 Development Tools for the .NET Framework 3.0 add-in for Visual Studio 2005 is installed on the system.
VS2005_WAP_PROJECT_INSTALLED	Indicates whether or not the Web Application

	Project template for Visual Studio 2005 is installed on the system. This project template is available as a standalone application and as a part of Visual Studio 2005 SP1.
VS2005_SP_LEVEL	Indicates the service pack level for Visual Studio 2005 Standard Edition and higher.
VSTF2005_SP_LEVEL	Indicates the service pack level for Visual Studio 2005 Team Foundation.
VB2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visual Basic 2005 Express Edition.
VC2005EXPRESS_SP_LEVEL	Indicates the service pack

	level for Visual C++ 2005 Express Edition.
VCSHARP2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visual C# 2005 Express Edition.
VJSHARP2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visual J# 2005 Express Edition.
VWD2005EXPRESS_SP_LEVEL	Indicates the service pack level for Visual Web Developer 2005 Express Edition.
DEXPLORE_2005_INSTALLED	Indicates whether or not the Document Explorer 2005 runtime components package is installed on the 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 editio 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 Basic 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 2008 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 Developer 2008 Express Edition.
VWD90EXPRESS_SP1	Indicates

	whether or not service pack 1 for Visual Web Developer 2008 Express Edition is installed.
DEXPLORE_2008_INSTALLED	Indicates whether or not the Document Explorer 2008 runtime components package is installed on the system.

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.

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.

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.

WixUI Dialog Library

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

Built-in 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 [light.exe](#) to reference the WixUIExtension. For example:

```
light -ext WixUIExtension -cultures:en-us Product.wixobj -c
```

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.r
```

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
WixUIExclamationlco	Exclamation icon on the WaitForCostingDlg	32 × 32
WixUIInfolco	Information icon on the cancel and error dialogs	32 × 32
WixUINewlco	Button glyph on the BrowseDlg	16 × 16
WixUIUplco	Button glyph on the BrowseDlg	16 × 16

Customizing the ExitDlg

The ExitDlg is the [dialog in the built-in WixUI dialog sets](#) that is displayed at the end of a successful setup. The ExitDlg supports showing both optional, customizable text and an optional checkbox.

See [How To: Run the Installed Application After Setup](#) 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 in
```

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:

```
<CustomAction Id="CA_Set_WIXUI_EXITDIALOGOPTIONALTEXT" Property="WI  
<InstallUISequence>  
  <Custom Action="CA_Set_WIXUI_EXITDIALOGOPTIONALTEXT" After="FindR  
</InstallUISequence>
```

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
```

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 [WixUI Dialog Library Reference](#).

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 [WixUI_InstallDir](#) dialog set, you would do the following:

1. 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.
3. 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="NewDialc
```

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="NewDi
```

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 [WixUI_InstallDir](#) 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" />  
<Publish Dialog="SpecialDlg" Control="Next" Event="NewDialog" />
```

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:

```
<Publish Dialog="WelcomeDlg" Control="Next" Event="NewDialog" />
```

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="NewDialog" />
```


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 Produ
```

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 Produ
light -ext WixUIExtension -cultures:fr-fr Product.wixobj -out Produ
light -ext WixUIExtension -cultures:de-de Product.wixobj -out Produ
light -ext WixUIExtension -cultures:it-it Product.wixobj -out Produ
light -ext WixUIExtension -cultures:ja-jp Product.wixobj -out Produ
light -ext WixUIExtension -cultures:pl-pl Product.wixobj -out Produ
light -ext WixUIExtension -cultures:ru-ru Product.wixobj -out Produ
light -ext WixUIExtension -cultures:es-es Product.wixobj -out Produ
```

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 Applicatio
      ...
    </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 Applicatio
```

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 [ALLUSERS](#) 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 [the WixUI dialog reference](#) for detailed descriptions of each of the above dialogs.

WixUI_FeatureTree Dialog Set

WixUI_FeatureTree is a simpler version of [WixUI_Mondo](#) 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 [the WixUI dialog reference](#) 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:

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

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 [the WixUI dialog reference](#) for detailed descriptions of each of the above dialogs.

WixUI_Mondo Dialog Set

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

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.
ErrorDlg	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

	<p>a checkbox and custom text to this dialog, see Customizing Built-in WixUI Dialog Sets and How To: Run the Installed Application After Setup.</p>
FatalError	<p>A dialog that displays a summary error dialog if setup fails.</p>
FeaturesDlg	<p>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.</p>
FilesInUse	<p>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.</p>
InstallDirDlg	<p>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.</p>
InstallScopeDlg	<p>A dialog that allows the user to choose to install the product for all users or for the current user.</p>
InvalidDirDlg	<p>A dialog that displays an error if the user selects an invalid installation directory.</p>
LicenseAgreementDlg	<p>A dialog that displays the end user license agreement and includes Back and Next buttons. Unlike the</p>

	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.

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 [Using WiX extensions](#) topic.

For information on how to use localized WiX extensions, please visit the [Localized extensions](#) topic.

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: [Code Access Security](#) 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:

1. 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.

Enabling IntelliSense for WiX extensions

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.

Localized Extensions

You can create your own localized extensions like [WixUIExtension](#) using [lit.exe](#). Localized extensions can even contain multiple languages. Products using these extensions can pass the -cultures switch to [light.exe](#) along with the -ext switch to reference the extension.

Authoring Libraries

[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.

Authoring Localization Files

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. For example, part of the WixUIExtension's en-US resources are copied below.

```
<?xml version="1.0" encoding="utf-8"?>
<WixLocalization Culture="en-us" xmlns="http://schemas.microsoft.co
  <String Id="WixUIBack" Overridable="yes">&Back</String>
  <String Id="WixUINext" Overridable="yes">&Next</String>
  <String Id="WixUICancel" Overridable="yes">Cancel</String>
  <String Id="WixUIFinish" Overridable="yes">&Finish</String>
  <String Id="WixUIRetry" Overridable="yes">&Retry</String>
  <String Id="WixUIIgnore" Overridable="yes">&Ignore</String>
  <String Id="WixUIYes" Overridable="yes">&Yes</String>
  <String Id="WixUINo" Overridable="yes">&No</String>
  <String Id="WixUIOK" Overridable="yes">OK</String>
</WixLocalization>
```

These [String](#) 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.

Building Libraries

When all the fragment authoring and localization files are complete, they can be compiled and linked together using [candle.exe](#) and [lit.exe](#).

First compile all the .wxs sources.

```
candle.exe example1.wxs -out example1.wixobj  
candle.exe example2.wxs -out example2.wixobj
```

Now link together all the .wixobj files and .wxl files for each culture you want available in the extension library.

```
lit.exe example1.wixobj example2.wixobj -loc en-us.wxl -loc de-de.wxl
```

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 [WixUIExtension](#) 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.

```
command.exe example.wxs -ext WixUIExtension -out example.wixobj  
light.exe example.wixobj -ext WixUIExtension -cultures:en-us -loc e
```

Patch Building

Patches are updates to a product or products. WiX supports two different ways of creating them:

[Using Patch Creation Properties](#) which requires that you have the Windows Installer 3.0 or newer SDK installed for full support of included examples.

[Using Purely WiX](#) which uses functionality provided in WiX and does not require additional tools.

There are also [restrictions](#) 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 [Patching and Upgrades](#).

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\Sample.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"
    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.
      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 Fe
      <ComponentRef Id="SampleComponent" />
    </Feature>
  </Fragment>

  <Fragment>
    <DirectoryRef Id="SampleProductFolder">
      <Component Id="SampleComponent" Guid="{C28843DA-EF08-41
        <File Id="SampleFile" Name="Sample.txt" Source=".\"$
      </Component>
    </DirectoryRef>
  </Fragment>

  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder" Name="PFiles">
        <Directory Id="SampleProductFolder" Name="Patch Sam
          </Directory>
        </Directory>
      </Directory>
    </Fragment>
  </Wix>
```

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"
  >

  <PatchInformation
    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"
    MediaSrcProp="Sample"
    Name="Sample"
    SequenceStart="5000">
    <UpgradeImage SourceFile="C:\sample\1.1\admin\product.m
      <TargetImage SourceFile="C:\sample\1.0\admin\produc
        Id="SampleTarget" IgnoreMissingFiles="no" />
      </UpgradeImage>
    </Family>

  <PatchSequence PatchFamily="SamplePatchFamily"
    Sequence="1.0.0.0"
    Supersede="yes" />

</PatchCreation>
</Wix>
```

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.

```
cd /d %~dp0\patch
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 [restrictions](#) 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\Sample.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"
    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.
      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 Fe
      <ComponentRef Id="SampleComponent" />
    </Feature>
  </Fragment>

  <Fragment>
    <DirectoryRef Id="SampleProductFolder">
      <Component Id="SampleComponent" Guid="{C28843DA-EF08-41
        <File Id="SampleFile" Name="Sample.txt" Source=".\"$
      </Component>
    </DirectoryRef>
  </Fragment>

  <Fragment>
    <Directory Id="TARGETDIR" Name="SourceDir">
      <Directory Id="ProgramFilesFolder" Name="PFiles">
        <Directory Id="SampleProductFolder" Name="Patch Sam
          </Directory>
        </Directory>
      </Directory>
    </Fragment>
  </Wix>
```

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' Super
      <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\d
```

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.

```


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 [restrictions](#) 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 uninstalleable patches

There are certain elements referenced in [restrictions](#) that cannot be added or modified if the patch is to be uninstalleable. 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 uninstalleable when you want it to be uninstalleable.

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 [Changing the Product Code](#).

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 [Patch Creation Properties](#) file using the [PatchMetadata/@AllowRemoval](#) attribute or into the [patch XML](#) file using the [Patch/@AllowRemoval](#) attribute.

Beside that, certain tables cannot be modified in the upgrade package from which a patch is built. Read [Uninstallable Patches](#) for the current list of tables. Pyro.exe will error if one of these tables would be modified when building a [patch XML](#) file.

The following table lists tables and corresponding elements or attributes in WiX.

Table	Element or Attribute
BindImage	File/@BindPath
Class	Class
Complus	Component/@ComPlusFlags
CreateFolder	CreateFolder
DuplicateFile	CopyFile
Environment	Environment
Extension	Extension
Font	File/@FontTitle

IniFile	IniFile
IsolatedComponent	IsolatedComponent
LockPermissions	Permission
MIME	MIME
MoveFile	CopyFile
ODBCAttribute	ODBCDriver/Property
ODBCDataSource	ODBCDataSource
ODBCDriver	ODBCDriver
ODBCSourceAttribute	ODBCDataSource/Property
ODBCTranslator	ODBCTranslator
ProgId	ProgId
PublishComponent	Category
RemoveIniFile	IniFile
SelfReg	File/@SelfRegCost
ServiceControl	ServiceControl
ServiceInstall	ServiceInstall
TypeLib	TypeLib

Verb

Verb

Major upgrade patches are not uninstalleable.

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 [Code Page Handling](#).

Setting the Code Page using WiX

Top-level elements like [Product](#), [Module](#), [Patch](#), and [PatchCreation](#) 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 .wxi 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 [Creating a Database with a Neutral Code Page](#) for more information.

For a walkthrough about how to author a build localized packages using WiX see [How To: Make your installer localizable](#) and [How To: Build a localized version of your installer](#).

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 [Windows Installer XML toolset](#) development community so I figured I should get some administrative details out of the way. If you are interested in contributing code to the Windows Installer XML toolset, it is very important to read through all four of these topics.

1) The Windows Installer XML toolset copyright is held by [Microsoft](#).

I want to be very up front about the copyright of the Windows Installer XML toolset and how it affects us as developers. Microsoft is the sponsor of the Windows Installer XML project. Before a contribution can be accepted into the WiX project, the lawyers have asked that we assign our rights to those contributions to Microsoft. By having developers sign a copyright assignment agreement, Microsoft can maintain single legal control of the project. That single legal control enables Microsoft to best defend the project in the future if there was ever any sort of legal challenge.

Before jumping to any conspiracy theories, please note that this copyright assignment is exactly the same process the [Free Software Foundation](#) has you go through if you work on a project they sponsor. Also, in Clause 5 of the Windows Installer XML assignment agreement your rights to your contribution are explicitly granted back to you. If you would like a copy of the assignment agreement, please contact wixadmin@microsoft.com.

2) The Windows Installer XML project is a [benevolent dictatorship](#).

In order to ensure consistency in the schema and maintain the quality of the tools, the Windows Installer XML project's CVS tree is locked down. In other words, commits to the code-base by the general populace are prevented. If you attempt commit changes, CVS will inform you that you have "Insufficient Karma to complete the task."

To have your contribution submitted to the project, please submit an assignment agreement as described above (you only need to do so

once) then send your code diff to WiX-devs@lists.sourceforge.net. The developers there will review the changes and someone will apply them to CVS as quickly as possible.

3) The Windows Installer XML community is a [meritocracy](#).

Those individuals in the community who demonstrate an understanding of the code base by actively participating on the [Windows Installer XML mailing lists](#) and consistently submitting high quality diffs will be given a “Karma boost”. With enough Karma you will earn the ability to commit changes directly to the Windows Installer XML project’s CVS tree.

Commit privileges should not be taken lightly. It is very important that the WiX toolset maintain a high quality bar because many people depend on the tools working properly. Very few developers earn these privileges. In fact, in over four years of development, only five developers have earned commit privileges to the internal Windows Installer XML project.

4) The Windows Installer XML developers are all [volunteers](#).

Everyone (to the best of my knowledge) that works on the Windows Installer XML toolset does so in his or her free time. Please keep that fact in mind when asking for help, submitting code diffs, or interacting with any members of the project. We all want to help to make the Windows Installer XML toolset as solid a tool as possible, but sometimes “real jobs” and “significant others” have to take a higher precedence.

If worse comes to worse, you have access to the source code. Try reading for a while. :)

Reprinted from

<http://blogs.msdn.com/robmen/archive/2004/04/14/112970.aspx>.

Copyright  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 [Windows SDK for Windows Server 2008 and .NET Framework 3.5](#) and/or Visual Studio 2008:

- 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.

[HTML Help SDK 1.4](#) 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 [Visual Studio Extensibility Center](#).

To install Votive on Visual Studio 2005 or 2008, 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\WinSDKNetFxT
hive="LocalMachine"
failonerror="false"/>
```

Replace this with the following element:

```
<readregistry
  property="sdkInstallRoot"
  key="SOFTWARE\Microsoft\Microsoft SDKs\Windows\v6.1\WinSDKNetFxTo
  hive="LocalMachine"
  failonerror="false"/>
```

Note the only difference is that the "v6.0a" changed to "v6.1" in the "key" attribute.

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 Conventions

Build File Format

In order to promote consistency 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 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 Simple 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 appropriate 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 extension 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 [Wix](#) 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/XmlSchemaExten
  <xs:attribute name="RequiredVersion" type="xs:string">
  <xs:annotation>
    <xs:documentation>
      The version of this extension required to compile the defin
    </xs:documentation>
    <xs:appinfo>
      <xse:parent namespace="http://schemas.microsoft.com/wix/200
    </xs:appinfo>
  </xs:annotation>
</xs:attribute>
</xs:schema>
```

Creating a Simple 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.WindowsInstallerXml` namespace.

```
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))]
```

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
```

Creating a Preprocessor Extension

The preprocessor in WiX allows extensibility 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 [Creating a Simple Extension](#) topic.

1. 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.WindowsInstallerXml namespace.

```
using Microsoft.Tools.WindowsInstallerXml;
```

3. Make your SamplePreprocessorExtension class implement PreprocessorExtension.

```
public class SamplePreprocessorExtension : PreprocessorExtension
```

4. Add your SamplePreprocessorExtension to your [previously created SampleWixExtension class](#) 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.

```
private SamplePreprocessorExtension preprocessorExtension;  
  
public override PreprocessorExtension PreprocessorExtension  
{  
    get  
    {  
        if (this.preprocessorExtension == null)  
        {  
            this.preprocessorExtension = new SamplePreprocessorExtension();  
        }  
        return this.preprocessorExtension;  
    }  
}
```



```
        }
        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.

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 [Visual Studio Extensibility Developer Center](#). 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 [detection key](#) 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.sln 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\test.bat [all|smoke|test name]
```

Running the tests from the command line with MSTest.exe

Run MSTest with the test binaries.

```
mstest.exe c:\delivery\Dev\wix\build\debug\x86\wixtests.dll
```

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.

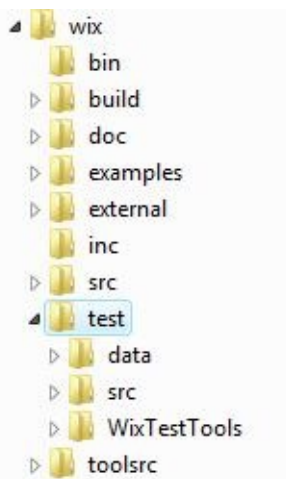
Location of the Tests

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

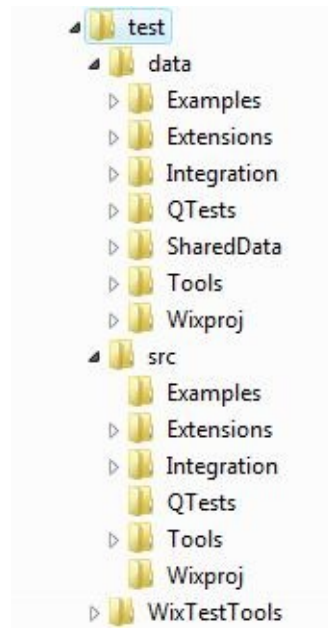
Integration: Tests for integration of two or more tools. Eg. Building an MSI from source with Candle and Light.

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 query the resulting 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");
}
```

Additional Resources

The following topics contain additional resources for the WiX toolset and Windows Installer:

[Getting Started Learning WiX](#)

[Useful Windows Installer Information](#)

[Getting Help](#)

Getting Started Learning WiX

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

How To Guides

This help file includes a set of [How To Guides](#) 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:

Introductory tutorials:

[Using the WiX Toolset to Integrate Setup into Your Development Process](#)
[Automate Releases With MSBuild And Windows Installer XML](#)

Comprehensive tutorial:

<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.

Audio-Visual

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/maillinglists.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 [WiX decompiler](#) (Dark) to convert your package into WiX authoring and then recompile it using the [WiX compiler](#) (Candle) and [WiX linker](#) (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.

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>

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. [Mailing lists](#) 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.