WiX tools and concepts MSBuild

MSBuild

WiX v4 is available as an MSBuild SDK. SDK-style projects have smart defaults that make for simple .wixproj project authoring. For example, here's a minimal .wixproj that builds an MSI from the .wxs source files in the project directory:

```
<Project Sdk="WixToolset.Sdk/4.0.2">
</Project>
```

TIP

SDK-style projects are easier to author but do work differently than "old" projects. For example, you might need to use explicit SDK imports to override the normal imports of SDK .props and .targets file. For more information, see the MSBuild documentation.

You can also create and edit SDK-style MSBuild projects in Visual Studio using FireGiant's HeatWave Community Edition.

INFO

See Signing packages and bundles for information about signing packages and bundles when using MSBuild.

Properties

You can set the following properties in your .wixproj to control the build:

Property	Description
AdditionalCub	Semicolon-delimited list of .cub files to use during MSI validation. Default: darice.cub for .msi packages; mergemod.cub for .msm packages
BindFiles	When **true**, bind referenced files into the output file. Valid only when building .wixlib WiX libraries. Default: **false**
CabinetCreationThreadCount	Specifies the number of simultaneous threads used when building multiple cabinets. Default: The number of logical processors in the system.
CompilerAdditionalOptions	A string specifying arbitrary Wix.exe command-line arguments to use during the build. Default: none
DebugType	Specifies the .wixpdb output: *full* for full symbol information or *none* to suppress the .wixpdb. Default: *full*
DefaultCompressionLevel	Specifies the compression level used when none is specified via `MediaTemplate` or `Media`. Valid values are: *none*, *low*, *medium*, *high*, *mszip*. Default: *medium*. Default Wix.exe switch: `-defaultcompressionlevel`
DefineConstants	Semicolon-delimited list of **name**=**value** string pairs that specify preprocessor variable values. Default: none

Property	Description
Ices	Semicolon-delimited list of ICE validation names to run during MSI validation. Default: all available ICEs
IncludeSearchPaths	Semicolon-delimited list of paths to use to locate `<?include?>` files. Default: current directory
InstallerPlatform	Architecture of the package or bundle. Valid values are: *x86*, *x64*, *arm64*. Default: `$(Platform)`. Default Wix.exe switch: `-arch`
IntermediateOutputPath	Path used for intermediate outputs. Default: obj/*platform*/*configuration*
LinkerAdditionalOptions	A string specifying arbitrary Wix.exe command-line arguments to use during the build. Default: none
OutputType	Specifies the type of package being built. Valid values are: *Package*, *Module*, *Patch*, *PatchCreation*, *Library*, *Bundle*, *IntermediatePostLink*. Default: *Package*
Pedantic	If **true**, turns on pedantic warning messages. Default: **false**
SuppressAllWarnings	If **true**, turns off all warning messages. Default: **false**
SuppressIces	Semicolon-delimited list of ICE validation names to *not* run during MSI validation. Default: none
SuppressSpecificWarnings	Semicolon-delimited list of warning message numbers to turn off. Default: none

Property	Description
SuppressValidation	If **true**, turns off MSI validation. Default: **false**
TreatSpecificWarningsAsErrors	Semicolon-delimited list of warning message numbers to treat as errors. Default: none
TreatWarningsAsErrors	If **true**, treats all warning messages as errors. Default: **false**
ValidationAdditionalOptions	A string specifying arbitrary Wix.exe command-line arguments to use during validation. Default: none
VerboseOutput	If **true**, turns on verbose messages. Default: **false**

Items

Item	Description
BindPath	Bind paths used to locate payload files. To create named bind paths, specify `BindName` metadata with the name of the bind path.
Compile	Files to compile. By default, the WiX SDK automatically includes all WiX authoring using the wildcard `**/*.wxs`. To control default items, see the project SDK documentation.
EmbeddedResource	Localization files used to build locale-specific packages. By default, the WiX SDK automatically includes all localization files using the wildcard `**/*.wxl`. To control default items, see the project SDK

Item	Description
	documentation.
WixLibrary	Paths to WiX libraries (.wixlib files) that contain authoring referenced by the package being built.

Project references

`ProjectReference` items to other projects are an MSBuild mechanism to ensure that a dependency project is built before the project that depends on it. For example, a .wixproj project depends on a .csproj project to ensure that the application to be installed is built before the .wixproj that installs it. The WiX MSBuild targets extend `ProjectReference`s to create bind paths and preprocessor variables that contain useful information about dependency projects.

NOTE

The WiX MSBuild targets create identifiers for bindpath and preprocessor variables from referenced projects. Characters that are invalid in those identifiers are replaced with underscores. (Identifiers begin with a letter or underscore and are optionally followed by alphanumeric characters, underscores, and/or periods.) For example, the space in `My Exe.csproj` will be replaced and `My_Exe` will be used as the bindpath and preprocessor variable name.

Bind paths

The WiX MSBuild targets create a bind path to the output directory of each referenced project. That means you can specify, for example, an .exe from a .csproj project using just the file name. For example:

```
  <File Source="ConsoleApp42.exe" />
```

The WiX MSBuild targets create a number of preprocessor variables for each referenced project.

Variable	Example	Example value
ProjectName.Configuration	$(MyProject.Configuration)	Release
ProjectName.FullConfiguration	$(MyProject.FullConfiguration)	Release|ARM64
ProjectName.Platform	$(MyProject.Platform)	ARM64
ProjectName.ProjectDir	$(MyProject.ProjectDir)	C:\source\repos\ConsoleApp42\
ProjectName.ProjectExt	$(MyProject.ProjectExt)	.csproj
ProjectName.ProjectFileName	$(MyProject.ProjectFileName)	MyProject.csproj
ProjectName.ProjectName	$(MyProject.ProjectName)	MyProject
ProjectName.ProjectPath	$(MyProject.ProjectPath)	C:\source\repos\ConsoleApp42\MyApp.csproj
ProjectName.TargetDir	$(MyProject.TargetDir)	C:\source\repos\ConsoleApp42\bin\Release\
ProjectName.TargetExt	$(MyProject.TargetExt)	.exe

Variable	Example	Example value
ProjectName.TargetFileName	$(MyProject.TargetFileName)	MyProject.exe
ProjectName.TargetName	$(MyProject.TargetName)	MyProject
ProjectName.TargetPath	$(MyProject.TargetPath)	C:\source\repos\ConsoleApp42\bin\Release\MyProject.exe
ProjectName.Culture.TargetPath	$(MyProject.en-US.TargetPath)	C:\source\repos\ConsoleApp42\bin\Release\en-US\MyProject.msi

As their name suggests, the following preprocessor variables are only available when building a `.sln` file. Building inside Visual Studio always uses the `.sln` file, so it can be a surprise that these preprocessor variables will not be available when using the command-line to build a project file.

Variable	Example	Example value
SolutionDir	$(SolutionDir)	C:\source\repos\MySolution\
SolutionExt	$(SolutionExt)	.sln
SolutionFileName	$(SolutionFileName)	MySolution.sln
SolutionName	$(SolutionName)	MySolution
SolutionPath	$(SolutionPath)	C:\source\repos\MySolution\MySolution.sln

Centralizing MSBuild properties and targets

Sometimes you need to add or modify several of the same properties in multiple MSBuild projects, like manufacturer name, copyright, product name, and so forth. Instead of editing every single project, you can manage properties from a central location in a file named Directory.Build.props.

INFO

Directory.Build.props is a feature of Microsoft.Common.props, which the WiX v4 MSBuild targets consume. The same is also true of Directory.Build.targets and Microsoft.Common.targets. You can read more about this support here.

To use Directory.Build.props, add it to the root of your project -- MSBuild will find the file in parent directories -- and give it a property group. For example:

```xml
<?xml version="1.0" encoding="utf-8"?>
<Project>
  <PropertyGroup>
    <MyProductNameProperty>My Fancy Productname</MyProductNameProperty>
  </PropertyGroup>
</Project>
```

You can then reference `MyProductNameProperty`, for example, in other properties:

```xml
<PropertyGroup>
  <Product>$(MyProductNameProperty)</Product>
</PropertyGroup>
```

To make property values available as preprocessor variables in your WiX authoring, add them to the `DefineConstants` property. For example:

```
<Project Sdk="WixToolset.Sdk/4.0.2">
  <PropertyGroup Label="Globals">
     <DefineConstants>MyProductNameProperty=$(MyProductNameProperty);</DefineConstants>
  </PropertyGroup>
</Project>
```

And then in your WiX authoring, you can use `$()` preprocessor syntax to refer to the MSBuild-property-turned-WiX-preprocessor-variable:

```
<Package Name="$(MyProductNameProperty)" ...
```

You can now modify the values of your properties in the `Directory.Build.props` file and all the properties in your solution's projects, including WiX projects will be updated.

Edit this page