FIREGIANT (/)

Products (/products/) Pricing (/services/) Documentation (/docs/) Blog (/blog/)

Contact Us (/contact-us/)

Home (/) / WiX Toolset v3 Tutorial (/docs/wix/v3/tutorial/)
/ COM, Expression Syntax, Miscellanea (/docs/wix/v3/tutorial/com-expression-syntax-miscellanea/)
/ Components of a Different Color

Components of a Different Color

Heat harvests data from various sources (folders, files, DLLs, ActiveX controls, performance counters) where the number of entries might be too large to author the corresponding WiX source file manually. Its primary intention is to be run once, to collect the data to be used

About FireGiant

Now you can install with confidence. FireGiant arms you with accurate support from the creators of WiX.

Because there are no questions we can't answer or problems we can't solve, you can unleash the full functionality of WiX without a doubt.

later in the usual maintenance of the source files, not to be part of a build environment where it is run on a changing set of input data over and over again. If you still want to use it in this second way, you have to be very careful to make sure that changes in the input don't introduce unwelcome effects (mostly breaking the component rules). Heat does have features that help you achieve this goal but be careful when using them.

The first mode of Heat will help us author the necessary WiX source for one or more folders full of files:

```
heat dir folder -cg SampleGroup -out SampleGroup.wxs
```

will harvest the folder specified recursively, creating a `Fragment` consisting of a `ComponentGroup` with the name specified with the `-cg` switch. The group will contain as many components as there are files, each component having a single file inside, as the rules dictate, and will assign uniquely generated component, directory and file identifiers that remain the same when regenerated on the same input set. GUIDs will not be generated (only a placeholder text) unless explicitely instructed to do so by the `-gg` switch.

In addition to the component group, Heat also generates fragments containing directory references (`DirectoryRef` tags) for each folder traversed recursively. The root directory harvested will have the identifier `TARGETDIR`, unless you specify your own name:

```
heat dir path -dr MyDirName -cg SampleGroup -out SampleGroup.wxs
```

Note that the automatically generated identifiers use this name as a seed. Changing the name will change all identifiers, with possible dire component-rule violation consequences if changed at an inappropriate time.

Yet another switch, `-srd` will suppress the identifier generation for the root folder specified. File components in the root folder will refer to their `Directory` as either `TARGETDIR` or the name specified in the `-dr` switch and there will be no separate `DirectoryRef` fragment for the root folder:

```
heat dir path -srd -dr MyDirName -cg SampleGroup -out SampleGroup.wxs
```

The second mode of the tool deals with a single file. If there are registry, COM and similar items to be collected from the file, they will all be extracted by Heat (the subject here was the Interop assembly for the Shell32.dll system library):

```
heat file file -cg SampleGroup -out SampleGroup.wxs
```

This results in a similar source file, with all details duly extracted:

```
<?xml version="1.0" encoding="utf-8"?>
<Wix xmlns="http://schemas.microsoft.com/wix/2006/wi">
  <Fragment>
    <ComponentGroup Id="SampleGroup">
      <Component Id="cmpA8B0842041500B0ACE61F7EFD0FBD893" Directory="dir0F6F75DF46D1BACE2233EC573E6D4AA9" Guid="PUT-GUID-HERE">
        <File Id="filDDAAB2C11E1E5AE4668D99216C3B5523" KeyPath="yes" Source="SourceDir\SampleHeat\Interop.Shell32.dll" />
        <RegistryValue Root="HKCR" Key="CLSID\{0A89A860-D7B1-11CE-8350-444553540000}\InprocServer32\1.0.0.0" Name="Class" Value="Shell32.ShellDispatchInprocClass" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0A89A860-D7B1-11CE-8350-444553540000}\InprocServer32\1.0.0.0" Name="Assembly" Value="Interop.Shell32, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0A89A860-D7B1-11CE-8350-444553540000}\InprocServer32\1.0.0.0" Name="RuntimeVersion" Value="v2.0.50727" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0A89A860-D7B1-11CE-8350-444553540000}\InprocServer32\1.0.0.0" Name="CodeBase" Value="file:///[#filDDAAB2C11E1E5AE4668D99216C3B5523]" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0A89A860-D7B1-11CE-8350-444553540000}\InprocServer32" Name="Class" Value="Shell32.ShellDispatchInprocClass" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0
A89A860-D7B1-11CE-8350-444553540000}\InprocServe
r32" Name="Assembly" Value="Interop.Shell32, Ver
sion=1.0.0.0, Culture=neutral, PublicKeyToken=nu
ll" Type="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0
A89A860-D7B1-11CE-8350-444553540000}\InprocServe
r32" Name="RuntimeVersion" Value="v2.0.50727" Ty
pe="string" Action="write" />
        <RegistryValue Root="HKCR" Key="CLSID\{0
A89A860-D7B1-11CE-8350-444553540000}\InprocServe
r32" Name="CodeBase" Value="file:///[#filDDAAB2C
11E1E5AE4668D99216C3B5523]" Type="string" Action
="write" />
        ...
      </Component>
    </ComponentGroup>
  </Fragment>

  <Fragment>
    <DirectoryRef Id="TARGETDIR">
      <Directory Id="dir0F6F75DF46D1BACE2233EC57
3E6D4AA9" Name="SampleHeat" />
    </DirectoryRef>
  </Fragment>

  <Fragment>
    <DirectoryRef Id="dir0F6F75DF46D1BACE2233EC5
73E6D4AA9" />
  </Fragment>
</Wix>
```

A few additional words have to be added here. People often wonder whether they should author all their registry and other changes into their installation package or let their installed components (DLLs, for instance) register themselves by adding the registry entries or carrying out other setup tasks when they are first run. The answer is clear: *never use self registration*. The Windows installer needs to be able to keep track of all registration and file changes of your product in order to be able to track versions, updates and to remove the product completely and reliably. Moving important data out of the jurisdiction of the Installer mechanism will jeopardize this and only cause problems to your users. Using self registration is very strongly discouraged by Microsoft (https://msdn.microsoft.com/en-us/library/aa371608%28v=vs.85%29.aspx) and should be avoided at all costs.

The third mode works through a Visual Studio or a compatible project file and creates references to all files of the specified type: *Binaries*, *Symbols*, *Documents*, *Satellites*, *Sources* or *Content*. For instance, with *Binaries* the tools collects all binary files to be built and deployed by the project, all put into their respective components and directories, ready to be included in your final WiX source with a simple reference:

```
heat project projectfile -pog:Binaries -cg SampleGroup -out SampleGroup.wxs
```

In our previous examples, we used Heat to generate a fragment for later inclusion into a complete setup package. We can also tell it to create a module or a standalone product instead. For smaller packages, the tool might actually author the bulk of the WiX source files and only some items (GUIDs, textual descriptions) have to be added manually.

```
heat ... -template:module -cg SampleGroup -out SampleGroup.wxs heat ... -template:product -cg SampleGroup -out SampleGroup.wxs
```

Typelibs are supported directly in WiX, there is no need to use Heat or any other tool to gather all their internal information:

```
<File Id="file.dll" Name="file.dll" KeyPath="yes">
    <TypeLib Id="YOURGUID-0BFE-4B1A-9205-9AB900C7D0DA" Language="0" />
</File>
```

‹ COM, Expression Syntax, Miscellanea (/docs/wix/v3/tutorial/com-expression-syntax-miscellanea/)
up (/docs/wix/v3/tutorial/com-expression-syntax-miscellanea/)
Expression Syntax › (/docs/wix/v3/tutorial/com-expression-syntax-miscellanea/expression-syntax/)

CONTACT US (/contact-us/) | BLOG (/blog/)

PRODUCTS (/products/)

- HeatWave (/docs/heatwave/)
- HeatWave Build Tools (/products/heatwave-build-tools/)
- FireGiant WiX v3 (/products/fg-wix/)

SERVICES (/services/)

- Support Programs (/services/support-programs/)
- Code Review (/services/code-review/)
- Phone Consultation (/services/phone-consultation/)
- Custom Development (/services/custom-development/)

DOCUMENTATION (/docs/)

- HeatWave (/docs/heatwave/)
- WiX v4 Tutorial (/docs/wix/tutorial/)
- WiX v3 Tutorial (/docs/wix/v3/tutorial/)
- FireGiant WiX v3 (/docs/fg-wix/)

- About Us (/about-firegiant/)
- Why WiX (/why-wix/)

› Leadership Team (/about-firegiant/leadership-team/)

› Privacy Policy (/about-firegiant/privacy-policy/)