WiX v4 is here! Let us help. (/services/conversion-offer/?utm_source=firegiant.com&utm_medium=Display&
utm_campaign=conversion_offer&utm_term=let+us+help&utm_content=announcement_bar)

FIREGIANT (/)

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

Contact Us (/contact-us/)

Fragments

Fragments offer a way to break up large installation
projects. Basically, a fragment is a wrapper put around a
WiX source unit, be it as simple or as complicated as we
like. Similarly to object files or libraries we use in program
development, a fragment can be linked into other product
installation packages. If you consider the two WiX tools
we have, Candle and Light like compiler and linker,

fragments, just like source code modules, can be compiled independently into object code (.wixobj files). Rebuilding of a large installation package can be sped up dramatically by using the usual makefile approach, only recompiling .wxs source files that have been changed since the last compilation and linking the .wixobj files, fresh and old, together to form the final package.

As we can't have such large and complex projects as samples in a tutorial, our SampleFragment (/system/files /samples/SampleFragment.zip) will be necessarily somewhat constructed. We reuse our first installer but we outsource some parts to separate fragments in order to illustrate our point. We removed the component that installs the user manual file, thus, the following reference will point to nowhere:

```
<Feature Id='Complete' Title='Foobar 1.0' Descri
ption='The complete package.' ...>
    <Feature Id='Documentation' Title='Descripti
on' Description='The instruction manual.' Level=
'1'>
        <ComponentRef Id='Manual' />
    </Feature>
</Feature>
```

We place the removed component into a file of its own, inside a `Fragment` tag. Here we only *refer to* the `Directory` we have already declared in our main source file because we can't declare the same thing in two

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.

Get Supported › (/services/support-programs

places. Anything that can be delegated into a fragment has its variant tag: to refer to a feature defined elsewhere, we use `FeatureRef,` to refer to a property, we use `PropertyRef.`

```xml
<?xml version='1.0' encoding='windows-1252'?>
<Wix xmlns='http://schemas.microsoft.com/wix/2006/wi'>
    <Fragment Id='FragmentManual'>
        <DirectoryRef Id='INSTALLDIR'>
            <Component Id='Manual' Guid='YOURGUID-574D-4A9A-A266-5B5EC2C022A4'>
                <File Id='Manual' Name='Manual.pdf' DiskId='1' Source='Manual.pdf' KeyPath='yes'>
                    <Shortcut Id="startmenuManual" Directory="ProgramMenuDir" Name="Instruction Manual" Advertise="yes" />
                </File>
            </Component>
        </DirectoryRef>
    </Fragment>
</Wix>
```

We could already compile and link these two files into an installation package. Note that we didn't need to add a single line to the source to make the two compilation units link together. The fact that we referenced a component from one file that was defined in the other one is enough for the linking to happen as we expect it. Referencing one element from a fragment opens up the whole fragment and makes all its elements immediately available. This

behavior is different from what we are used to with the usual programming languages: fragment linking not only makes it possible to use elements defined elsewhere, using at least one of them will always switch on *everything* from the fragment referenced.

To build the sample, use the commands:

```
candle.exe SampleFragment.wxs Manual.wxs
light.exe -out SampleFragment.msi SampleFragmen
t.wixobj Manual.wixobj
```

Fragments have many uses not only inside a single setup project but to share common items between different projects as well. For instance, if you have several related applications that share one or more common files (eg. device drivers or other functionality in DLLs), you might need to ensure that removing one application will not remove the common file that is still needed by the other one.

In this case, put your common file into a fragment of its own and refer to this fragment from all of your separate application setups. As the component (and, therefore, the component GUID) is the same, Windows Installer will be able to keep track of all the application requiring the common files, including properly enforcing versioning rules so that a newer version will never be overwritten by an older one.

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/)

© 2023 FIREGIANT (/)

> About Us (/about-firegiant/)

> Why WiX (/why-wix/)

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

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