This is documentation for WiX Toolset **v3**, which is no longer actively maintained.

For up-to-date documentation, see the **latest version** (v4.0).

Version: v3

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 Application Name"/>
    </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\MyApplication.exe" KeyPath="yes"
Checksum="yes"/>
    </Component>
    <Component Id="documentation.html" Guid="PUT-GUID-HERE">
        <File Id="documentation.html" Source="MySourceFiles\documentation.html" KeyPath="yes"/>
    </Component>
</DirectoryRef>
```

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. If you do not set the KeyPath attribute, WiX will look at the child elements under the component in sequential order and try to automatically select one of them as a key path. Allowing WiX to automatically select a key path can be dangerous because adding or removing child elements under the component can inadvertantly cause the key path to change, which can lead to installation problems. In general, you should always set the KeyPath attribute to yes to ensure that the key path will not inadvertantly change if you update your setup authoring in the future.

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" Language="1033" Name="My Application
Name" Manufacturer="My Manufacturer Name">
        <Package InstallerVersion="300" Compressed="yes"/>
        <Media Id="1" Cabinet="myapplication.cab" EmbedCab="yes" />

        <!-- Step 1: Define the directory structure -->
        <Directory Id="TARGETDIR" Name="SourceDir">
            <Directory Id="ProgramFilesFolder">
                <Directory Id="APPLICATIONROOTDIRECTORY" Name="My Application Name"/>
            </Directory>
        </Directory>

        <!-- Step 2: Add files to your installer package -->
        <DirectoryRef Id="APPLICATIONROOTDIRECTORY">
            <Component Id="myapplication.exe" Guid="PUT-GUID-HERE">
                <File Id="myapplication.exe" Source="MySourceFiles\MyApplication.exe" KeyPath="yes"
Checksum="yes"/>
            </Component>
            <Component Id="documentation.html" Guid="PUT-GUID-HERE">
                <File Id="documentation.html" Source="MySourceFiles\documentation.html" KeyPath="yes"/>
            </Component>
        </DirectoryRef>

        <!-- Step 3: Tell WiX to install the files -->
        <Feature Id="MainApplication" Title="Main Application" Level="1">
            <ComponentRef Id="myapplication.exe" />
            <ComponentRef Id="documentation" />
        </Feature>
    </Product>
</Wix>
```

Edit this page