**Version: v3**

# 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.dll"/>
            <File Id="DEC2006_d3dx9_32_x86.cab"
                  Source="MySourceFiles\DirectXMinInstall\DEC2006_d3dx9_32_x86.cab"/>
        </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</ProgressText>
</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.

Edit this page