Quiet execution custom actions

The WixQuietExec custom action family allows you to run an arbitrary command line without showing any UI -- like the console windows that would otherwise appear appear when invoking the executable via Windows Installer's built-in custom action types. WixQuietExec and friends are included in WixToolset.Util.wixext.

Architecture-specific custom action DLLs

As noted in *Frequently-asked questions about upgrading from WiX v3 to WiX v4*, some extra effort is sometimes required due to WiX v4's support for architecture-specific custom actions. That effort for WixQuietExec and friends is a little bit different:

- The custom action ids are architecture-neutral (that is, they don't have architecture-specific suffixes).
- The custom action DLLs have architecture-specific ids.

The preprocessor is your friend: The `$(sys.BUILDARCHSHORT)` preprocessor variable contains the suffix you need for the custom action DLL id based on the architecture of the package you're building. For example, you can use `BinaryRef="Wix4UtilCA_$(sys.BUILDARCHSHORT)"` in your `CustomAction` authoring.

Quiet running as immediate custom actions

When WixQuietExec is run as an immediate custom action, it executes the command stored in the WixQuietExecCmdLine property. The following is an example of authoring an immediate WixQuietExec custom action:

```
<SetProperty
  Id="WixQuietExecCmdLine"
  Value="&quot;[%ComSpec]&quot; /c ECHO %NUMBER_OF_PROCESSORS%"
  Before="ConvolutedCoreCountCustomAction"
  Sequence="execute"
  />
<CustomAction
  Id="ConvolutedCoreCountCustomAction"
  BinaryRef="Wix4UtilCA_$(sys.BUILDARCHSHORT)"
  DllEntry="WixQuietExec"
  Execute="immediate"
  Return="check"
  />
...
<InstallExecuteSequence>
  <Custom Action="ConvolutedCoreCountCustomAction" Before="AppSearch" />
</InstallExecuteSequence>
```

If the exit code of the command line is nonzero, WixQuietExec returns ERROR_INSTALL_FAILURE. In this example, the setup fails because `Return="check"`. Changing the Return attribute value to `ignore` means the failure will be shown in the log (as normal) but execution will continue instead of failing.

WixQuietExec logs the command line and any console output from the processes it executes. For example:

```
WixQuietExec:  "C:\WINDOWS\SysWOW64\cmd.exe" /c ECHO %NUMBER_OF_PROCESSORS%
WixQuietExec:  16
```

Quiet running as deferred custom actions

When WixQuietExec is run as a deferred custom action, it executes the command stored in the custom action data -- which is the value of the property with the same id as the custom action. For example:

```
<SetProperty
  Id="ConvolutedWhoAmICustomAction"
  Value="&quot;[%ComSpec]&quot; /c WHOAMI"
  Before="ConvolutedWhoAmICustomAction"
  Sequence="execute"
  />
<CustomAction
  Id="ConvolutedWhoAmICustomAction"
  BinaryRef="Wix4UtilCA_$(sys.BUILDARCHSHORT)"
  DllEntry="WixQuietExec"
  Execute="deferred"
  Impersonate="no"
  Return="check"
  />

<InstallExecuteSequence>
  <Custom Action="ConvolutedWhoAmICustomAction" After="InstallInitialize" />
</InstallExecuteSequence>
```

If the exit code of the command line is nonzero, WixQuietExec returns ERROR_INSTALL_FAILURE. In this example, the setup fails because `Return="check"`. Changing the Return attribute value to `ignore` means the failure will be shown in the log (as normal) but execution will continue instead of failing.

WixQuietExec logs the command line and any console output from the processes it executes. For example:

```
WixQuietExec:  "C:\WINDOWS\SysWOW64\cmd.exe" /c WHOAMI
WixQuietExec:  nt authority\system
```

Silent running

As noted earlier, WixQuietExec's primary purpose is to hide any UI from launched processes. Also as noted above, WixQuietExec logs the console output from those processes. If you want to hide the UI *and* hide the console output from being added to the log, use WixSilentExec and WixSilentExecCmdLine instead of WixQuietExec and WixQuietExecCmdLine:

```
<SetProperty
  Id="WixSilentExecCmdLine"
  Value="&quot;[%ComSpec]&quot; /c ECHO %NUMBER_OF_PROCESSORS%"
  Before="ConvolutedCoreCountCustomAction"
  Sequence="execute"
  />
<CustomAction
  Id="ConvolutedCoreCountCustomAction"
  BinaryRef="Wix4UtilCA_$(sys.BUILDARCHSHORT)"
  DllEntry="WixSilentExec"
  Execute="immediate"
  Return="check"
  />

<InstallExecuteSequence>
  <Custom Action="ConvolutedCoreCountCustomAction" Before="AppSearch" />
</InstallExecuteSequence>
```

WixSilentExec supports silent and deferred execution like WixQuietExec.

64-bit awareness

WixQuietExec64 and WixSilentExec64 are variants of WixQuietExec and WixSilentExec that are 32-bit custom actions (regardless of name) that are 64-bit aware. Specifically, they know how to turn off file-system redirections (for example) to support running 64-bit processes. WiX v4 has native custom actions for all three supported platforms (x86, x64, and Arm64), so if you need to call a 64-bit process from a 32-bit package, use those by hard-coding the `BinaryRef`. For example:

```
<CustomAction
  Id="MyDeferredCustomAction"
  BinaryRef="Wix4UtilCA_X64"
  DllEntry="WixQuietExec"
  Execute="deferred"
  Impersonate="no"
  Return="check"
  />
```

WixQuietExec64 and WixSilentExec64 will be removed in the next major version of WiX.

Edit this page