MSI Tip: Building Releases with the InstallShield Automation Interface

Robert Dickau
Principal Technical Training Writer
Flexera Software

Abstract

The InstallShield Automation interface enables you to modify your project's properties outside the InstallShield IDE, as well as to perform operations such as building releases and patch configurations. This article gives some examples of how to build releases from a VBScript file using the ISWiProductConfig and ISWiRelease objects.

The ISWiProductConfig and ISWiRelease Objects

When working with the InstallShield environment, you can use the Release Wizard to create product configurations and release names, and to set such release properties as the media type, whether to include the Windows Installer redistributables, and whether to compress your data files. You can view your existing releases in the Releases view, and rebuild a release by right-clicking its icon in the Releases view and selecting Build, or pressing **F7** to rebuild the current release.

Inside your project file, each product configuration is represented by an ISWiProductConfig Automation object, and every release is represented by an ISWiRelease object. Moreover, there are collections called ISWiProductConfigs and ISWiReleases that store all the product configurations and releases in your project.

For example, you can place the following code in a VBScript file called EnumReleases.vbs, and execute the file to view a message box displaying all the product configurations and releases in a given project. (When opening a project with the Automation interface, note that the project should not be open in the InstallShield environment.)

```
' NOTE: ProgID changes with each InstallShield version;
' check documentation to see your version's ProgID
Set oISM = CreateObject("ISWiAuto14.ISWiProject")

' open project as read-only
oISM.OpenProject "D:\MySetups\BuildMe.ism", True

strAllReleases = ""

' walk through all product configurations
For Each oConfig in oISM.ISWiProductConfigs

    strAllReleases = strAllReleases + vbNewLine + oConfig.Name + vbNewLine

    ' walk through all releases in current product configuration
    For Each oRelease in oConfig.ISWiReleases

      strAllReleases = strAllReleases + vbTab + _ oRelease.Name + vbNewLine

    Next ' oRelease
```

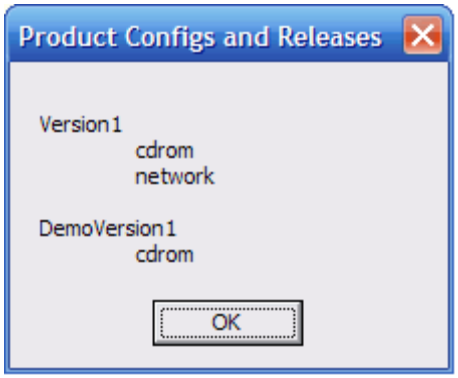
```
Next ' oConfig

MsgBox strAllReleases, , "Product Configs and Releases"

oISM.CloseProject
```

Listing 1: EnumReleases.vbs

When you run EnumReleases.vbs, you should see a message box similar to the following, showing all the product configurations, and, indented under each product configuration name, each release name.



To build a release inside a given product configuration, call the Build method of the specific ISWiRelease object. For example, placing the following code in a file called BuildRelease.vbs will build a release called "cdrom" contained in a product configuration called "Version1".

```
Set oISM = CreateObject("ISWiAuto14.ISWiProject")

' open project as read-only
oISM.OpenProject "D:\MySetups\BuildMe.ism", True

' build the release
oISM.ISWiProductConfigs("Version1").ISWiReleases("cdrom").Build( )

' for testing purposes
MsgBox "Done!"

oISM.CloseProject
```

Listing 2: BuildRelease.vbs

Combining the two previous examples, you can build every release contained in your project.

```
Set oISM = CreateObject("ISWiAuto14.ISWiProject")

' open project as read-only

oISM.OpenProject "D:\MySetups\BuildMe.ism", True

' walk through all product configurations
For Each oConfig in oISM.ISWiProductConfigs

    ' walk through all releases in current product configuration
    For Each oRelease in oConfig.ISWiReleases

      oRelease.Build( )
```

```
    ' for testing purposes
    MsgBox "Built " & oConfig.Name & "/" & oRelease.Name & "..."

Next ' oRelease

Next ' oConfig

' for testing
MsgBox "Done!"

oISM.CloseProject
```

Listing 3: BuildAll.vbs

For more information, see the InstallShield help topic "Automation Interface".

Creating a Release

You can also use the Automation interface to add product configurations and releases to your project, using the AddProductConfig and AddRelease methods. After adding a product configuration or release, you specify its properties (the properties you would ordinarily enter in the Release wizard) using the individual Automation properties of the ISWiProductConfig or ISWiRelease object you created.

For example, suppose you want to create a release called "new" inside an existing product configuration called "DemoVersion1". You can use the AddRelease method in your code as follows. Note that the script will fail if the release you name already exists in the project.

```
Set oISM = CreateObject("ISWiAuto14.ISWiProject")

' this time, open project as read-write
oISM.OpenProject "D:\MySetups\BuildMe.ism", False

' get existing configuration
Set oMyConfig = oISM.ISWiProductConfigs("DemoVersion1")

' add new release to existing configuration
Set oNewRelease = oMyConfig.AddRelease("new")

' set release properties
oNewRelease.Compressed = True
oNewRelease.SetupEXE = True
oNewRelease.TargetOS = 3 ' i.e., os9xNT: include both engines
' ...set other properties...

' if you want, build the new release
oNewRelease.Build( )

' for testing
MsgBox "Done!"

' this time, save the project before closing it
oISM.SaveProject
oISM.CloseProject
```

Listing 4: AddRelease.vbs

For information about other product configuration and release properties you can set with the Automation interface, see the InstallShield help topics "ISWiProductConfig Object" and "ISWiRelease Object".

Conclusion

This article only scratches the surface of what you can do with the InstallShield Automation interface: you can query and modify your project's actions and sequences, Summary Information stream, features, components, file links, and much more. For more information and examples, see the online help topic "Advanced Features > Automation Interface".