This partial chapter is an excerpt from the **Learning MSI Projects using InstallShield 2015** training manual. For information about purchasing this training manual and to learn about other Flexera Software training manuals, please visit http://www.flexerasoftware.com.

4

Building Releases

This chapter contains covers information on building releases, including:

- Using the Release Wizard to build release images
- Building releases from the command line
- Automating builds
- Creating a standalone build system
- Using the MSBuild option
- Integrating with source control
- Using support files

Release Wizard

The Release Wizard generates Windows Installer packages for distribution from a network, CD-ROM or DVD-ROM, or Web site. You run the Release Wizard to make distribution images; you can also run the Release Wizard to rebuild a project after changing its properties in the IDE.

To launch the Release Wizard, click the **Release Wizard** toolbar button, pull down the **Build** menu and select **Release Wizard**, or right-click the **Release** icon in the **Releases** view of the IDE. After viewing the **Welcome** panel, click **Next** to specify the properties of the current release.

The **Product Configuration** panel of the Release Wizard is where you specify a name for the current build configuration.

The product configuration name is used as the name of a folder in which release images are placed. By changing the configuration name from a previously defined build, the earlier build will be preserved.

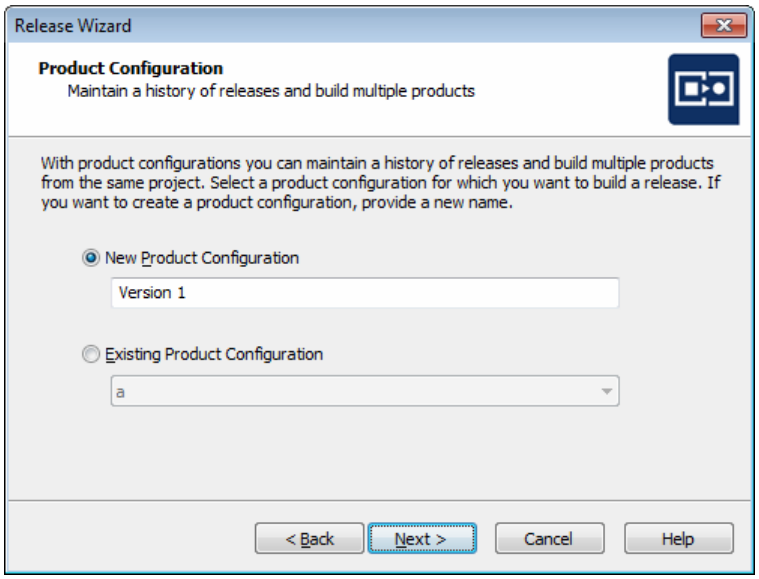

Figure 4-1: Release Wizard: Product Configuration

The **Specify a Release** panel is where you specify a name for the current release. Like the product configuration, the release name is the name of a folder (inside the product configuration folder) in which the Release Wizard places the release image it builds.

The release name is also the collective name for all the Release Wizard settings you enter in the following panels. To rebuild a particular release using its existing settings, select the name of the existing release.

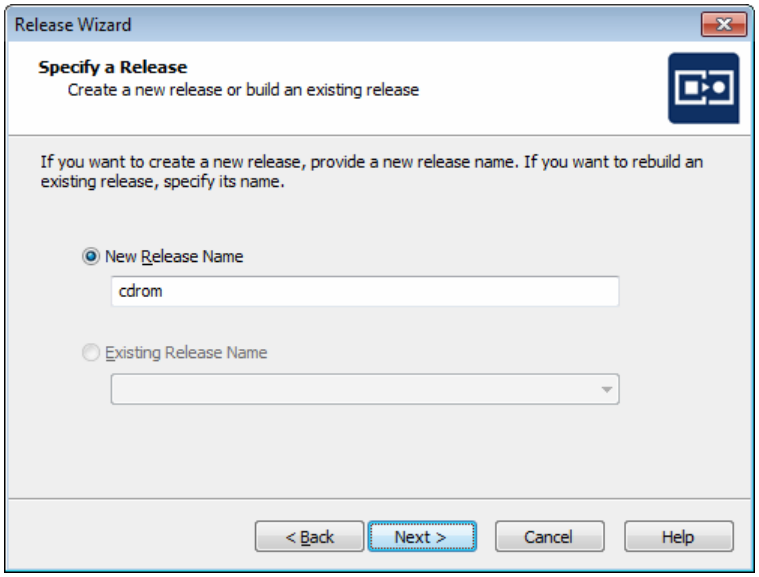

Figure 4-2: Release Wizard: Specify a Release

You can build some or all of the releases under a particular release configuration in the **Releases** view of the IDE by right-clicking the desired release configuration icon, selecting **Batch Build**, and then selecting the check boxes next to the desired release names.

The configuration and release names are simply two subfolders that define where the build image is stored. By changing either, the built image is stored in a different location—preserving previous builds in other folders. Rebuilding an already defined configuration and release overwrites the existing image.

The **Filtering Settings** panel allows you to specify features and components to include in the current release.

If you gave any features in the current project a value in their **Release Flags** property, that feature's files and data will be built into the current release only if you specify the flag in the **Release Flags** field of this panel.

For example, if you set a release flag called `FULLVERSION` on only one feature in the **Setup Design** (or **Features**) view, only that feature's files and data will be built into the current release if you specify `FULLVERSION` in the **Release Flags** field. If you do not specify a release flag in this panel, no feature filtering will be performed and all features will be included in the release.

Similarly, you can specify the languages on which you want to filter application data during the build. If you specified particular languages in a component's **Languages** property, the component's files and data will be built into the current release only if you check the appropriate language in the **Application Data Language Settings** field.

A component's Languages property has no effect on run-time component filtering. To handle run-time component filtering, for example, you can create a component condition that uses the predefined MSI property `ProductLanguage`, as described in Chapter 9, User Interface.

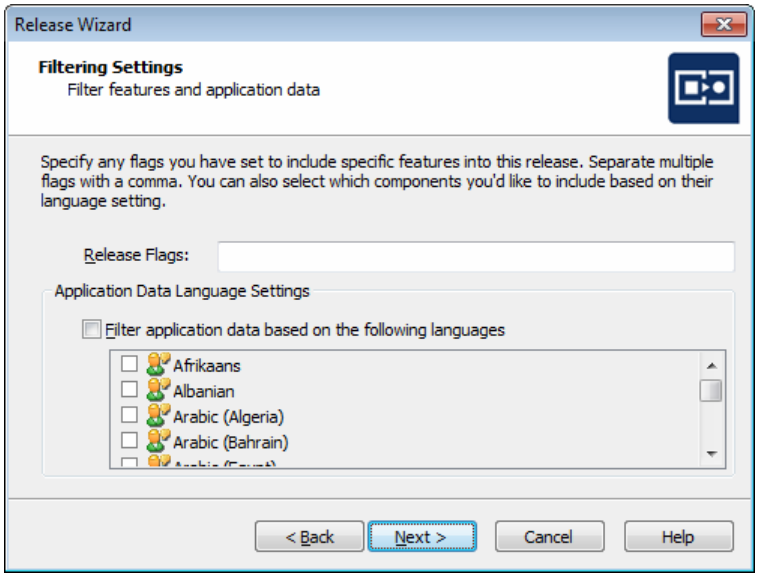

Figure 4-3: Release Wizard: Filtering Settings

Tip • *While release flags are used primarily to filter features at build time, InstallShield defines a property called `ISReleaseFlags`, which is set to a comma-separated list of release flags used in the current build. If you want, you can use this property at run time in conditions, as described in Chapter 8, Conditions and Actions.*

The **Setup Languages** panel allows you to specify which user interface languages you want the current release to support. You can also specify whether to display the setup language dialog box to the end user, and indicate which language you want the setup program to use as the default. Localization issues are discussed in Chapter 9, User Interface.

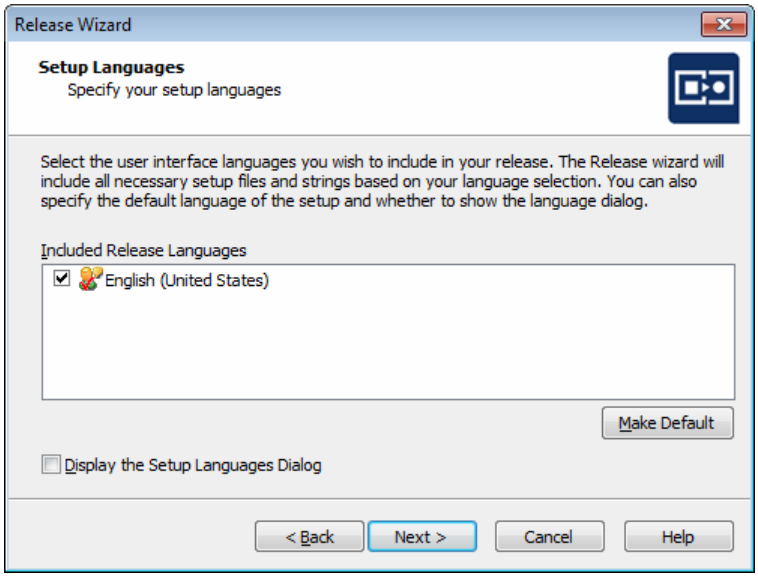

Figure 4-4: Release Wizard: Setup Languages

The **Media Type** panel is where you specify for what type of media—CD-ROM, DVD-ROM, network image, and so forth—you want to build the current release. The media type you select affects the size of each disk image folder (Disk1, Disk2, etc.) created by the Release Wizard.

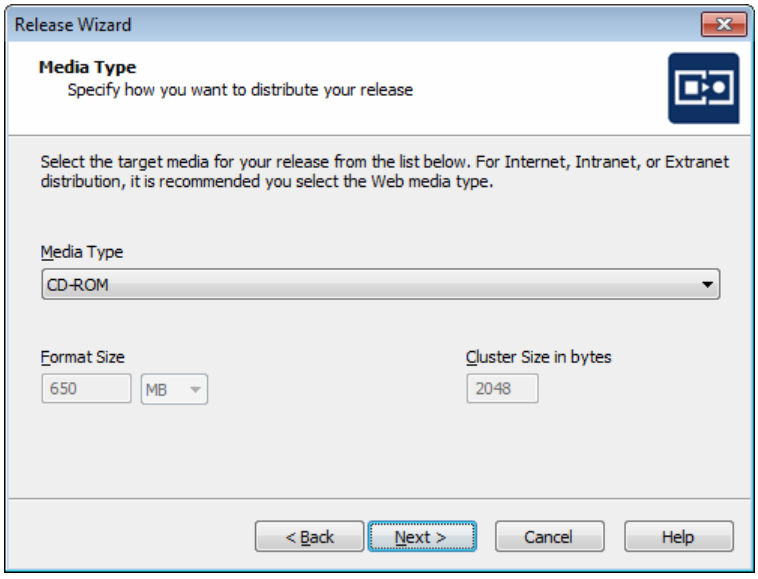

Figure 4-5: Release Wizard: Media Type

For media types that do not appear in the list, you can select the **Custom** media type, entering the capacity and cluster size for the custom media type.

In the **Disk Spanning Options** panel, select **Automatic** to specify that the Release Wizard should automatically distribute your data files across multiple disk images (assuming the installation program does not fit on a single disk), or select **Custom** to manually specify the disk image in which a given feature's files should be placed.

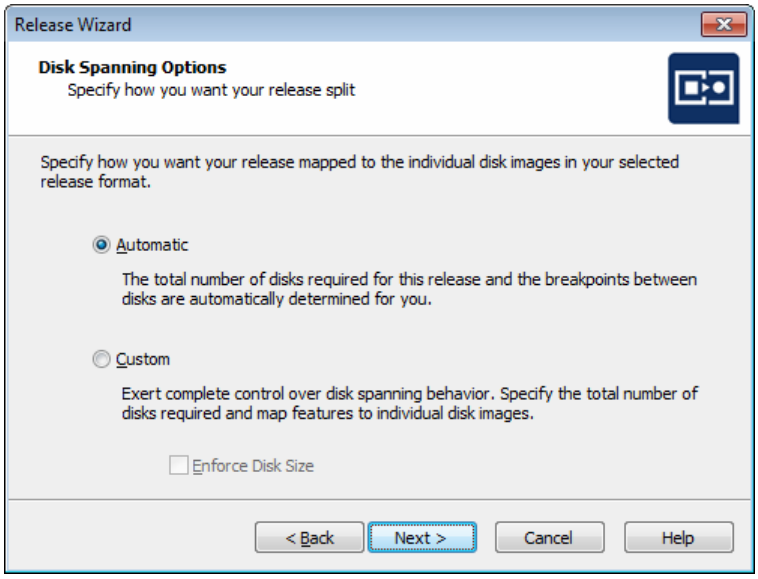

Figure 4-6: Release Wizard: Disk Spanning Options

The **Release Configuration** panel allows you to specify whether to compress all or some of the files included in the current release.

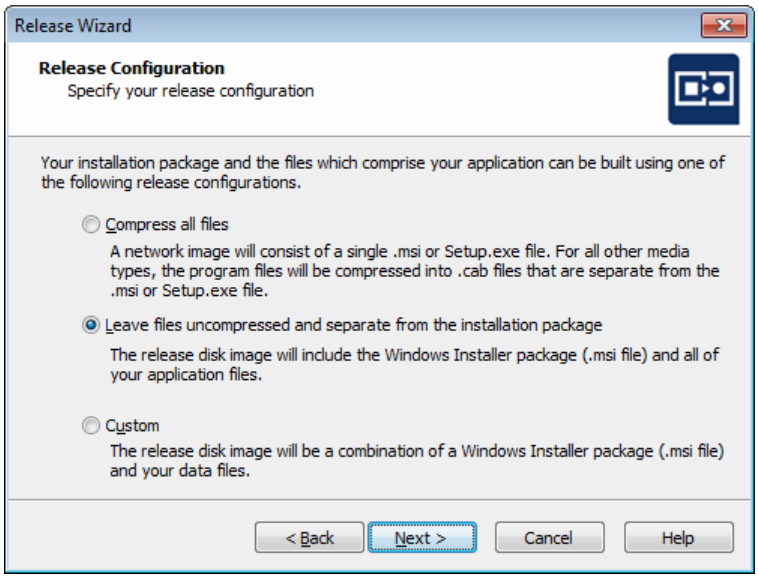

Figure 4-7: Release Wizard: Release Configuration

If you choose to compress files for a CD-ROM build, the compressed data files will be placed in cabinet (CAB) files outside `Setup.exe`. To compress data files inside `Setup.exe`, you can select the **Network** media type or a **One Executable Web build**.

If you choose **Custom**, you can specify which features you want to compress. You can specify whether to create one .cab file per component or one .cab file per feature.

In the **Setup Launcher** panel, specify whether to include the Windows Installer 3.1 redistributable with your installation. All Windows versions later than Windows 2000 ship with a version of the Windows Installer service, though in some cases it might be desirable to update it. In general, settings introduced in a given version of Windows Installer are ignored in earlier versions. For example, Windows Vista-specific settings are ignored on systems running an MSI version earlier than 4.0.

Note • *Windows Installer versions later than 3.1 are not listed on the* ***Setup Launcher*** *panel. Windows Installer 4.0 works only on Windows Vista systems, on which it comes pre-installed. You can install Windows Installer 4.5 in the Prerequisites view, described in Chapter 3, Installing Files.*

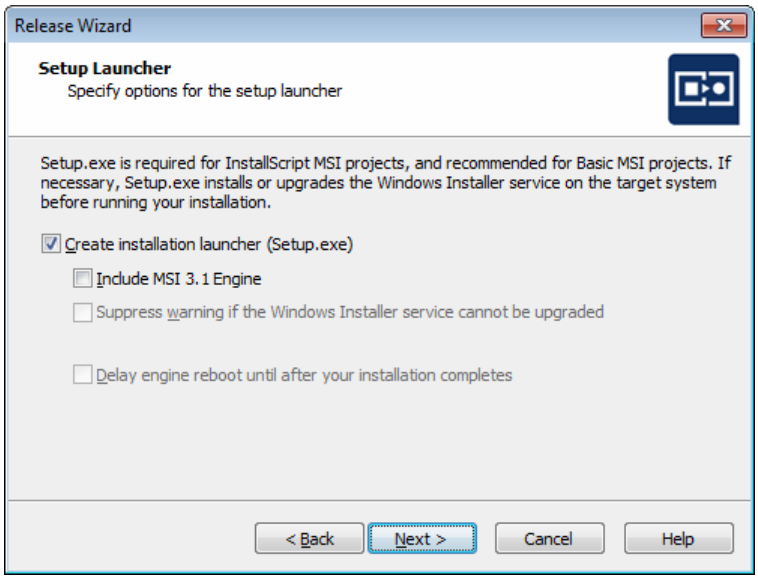

Figure 4-8: Release Wizard: Setup Launcher

The Setup.exe installation launcher is required if you include one or more of the Windows Installer redistributables, any InstallShield prerequisites, or include multiple languages in your project and build settings.

Tip • *InstallShield defines a run-time property called* `SETUPEXEDIR`*, which contains the directory from which* `Setup.exe` *was launched. For any release in which the MSI database is stored inside* `Setup.exe`*, this property will be different from the run-time* `SourceDir` *property, which specifies the directory from which the MSI database is running. If the MSI database is stored inside* `Setup.exe`*,* `SourceDir` *will refer to the temporary directory into which the MSI database is uncompressed.*

Windows Installer 3.x requires that the target operating system be Windows Server 2003, Windows XP, or Windows 2000 Service Pack 3.

If you specified that the Windows Installer 3.1 redistributable should be included with the current release image, the **Windows Installer Location** panel lets you specify where the redistributable is in relation to the disk images.

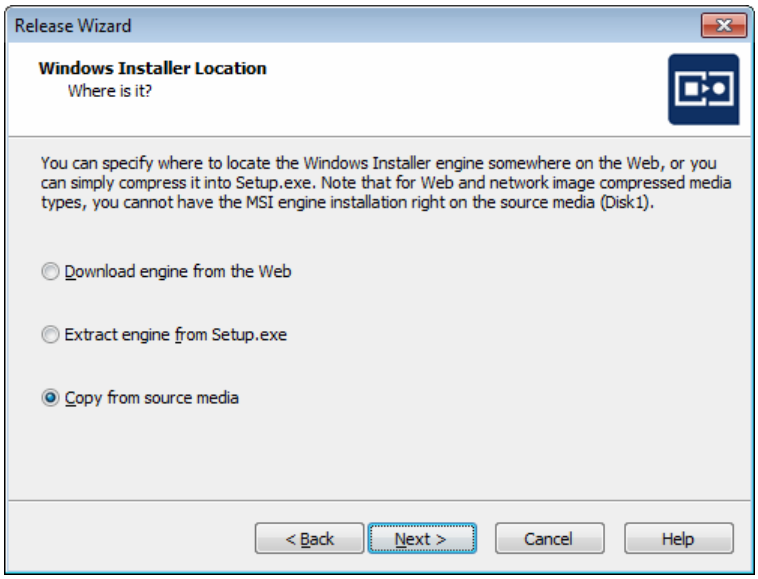

Figure 4-9: Release Wizard: Windows Installer Location

You can choose any of the following locations:

- **Download engine from the Web**—Select this option to download the Windows Installer engine installer from the URL that is specified in the **Release** property sheet's **MSI 3.1 Engine URL** setting.
- **Extract engine from Setup.exe**—The Windows Installer redistributable will be placed inside `Setup.exe`.
- **Copy from source media**—The Windows Installer redistributable will be left uncompressed at the root of the build location.

If you selected any of the InstallShield prerequisites, the **InstallShield Prerequisites** panel lets you specify additional parameters.

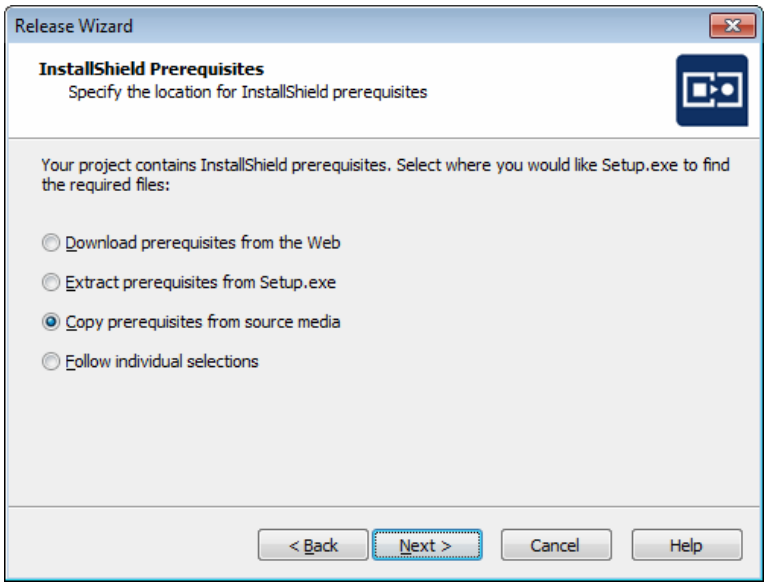

Figure 4-10: Release Wizard: Prerequisites Location

You can choose any of the following options:

- **Download prerequisites**—Select this option to download the prerequisite at install time. The download location is specified within the definition of the individual prerequisite.
- **Extract prerequisite from Setup.exe**—Select this option to extract the prerequisite from the setup executable itself. This option requires that the prerequisites be available locally when the image is built.
- **Copy prerequisites from source media**—The prerequisites will be left uncompressed in the install image. This option requires that the prerequisites be available locally when the image is built.
- **Follow individual selections**—The location specified within each prerequisite is used.

In the **Digital Signature** panel, you can specify digital signature information for your package and the files in your package.

If you specify digital signature information, the **Digital Signatures Options** panel is the next panel displayed in the Release Wizard. This panel is where you specify which files in your installation should be digitally signed at build time.

All executable files (including `.exe`, `.dll`, `.ocx`, `.sys`, `.cpl`, `.drv`, and `.scr` files) in an installation must be digitally signed for the Certified for Windows program.

InstallShield 2015 introduced several improvements for digitally signing your installation and application files at build time including support for SHA-256 certificates. In addition, InstallShield can now reference a certificate store that contains the certificate that you want to use as an alternative to specifying a `.pfx` file on your machine.

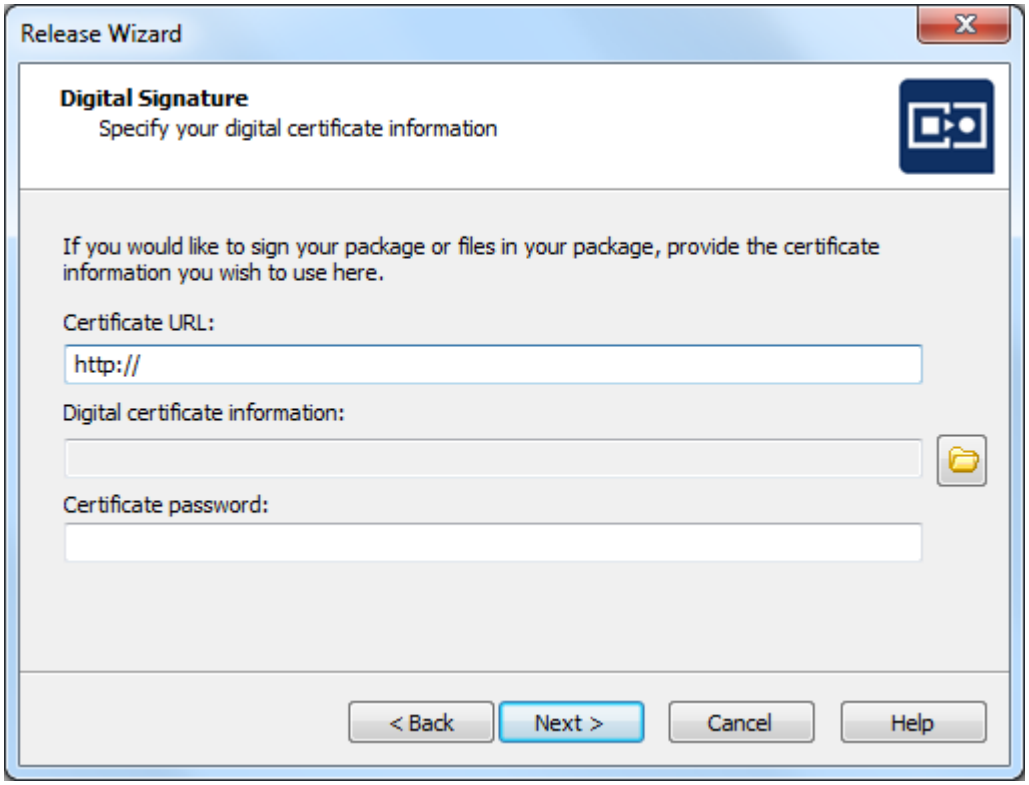

Figure 4-11: Release Wizard: Digital Signature

Tip • *For more information about digital signatures, refer to "Digital Signatures and Security" in the InstallShield Help Library.*

If you specified to create the `Setup.exe` installation launcher, you can specify a password that the user must enter before running the installation, and can specify your own copyright information for the `Setup.exe` executable.

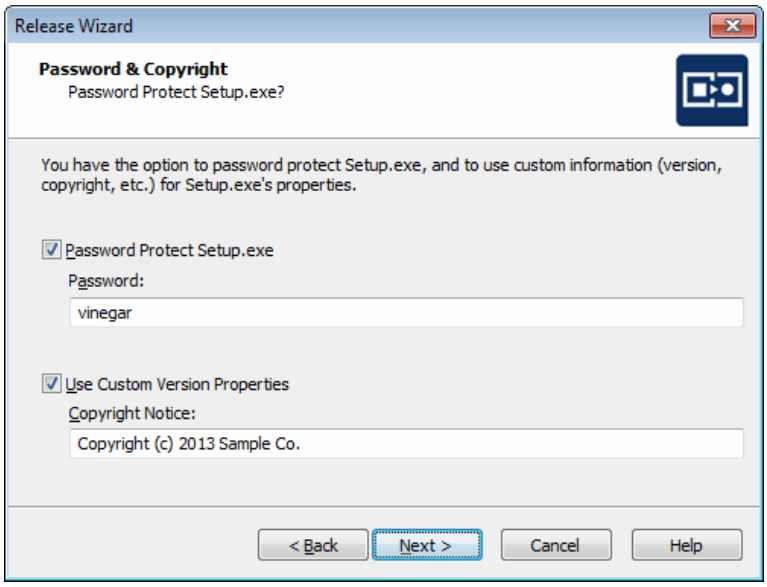

Figure 4-12: Release Wizard: Password & Copyright

In the **.NET Framework** panel, you can specify whether to include the .NET Framework redistributable with your disk images.

If you select to include the .NET Framework, the choices for its location are the same as for the Windows Installer redistributables described above.

You can also specify the version of the .NET Framework that you want to be included with your release image. For .NET Framework 2.0, 1.1, or 1.0 redistributables (32-bit), configure the .NET settings through the **.NET Framework** panel of the Release Wizard.

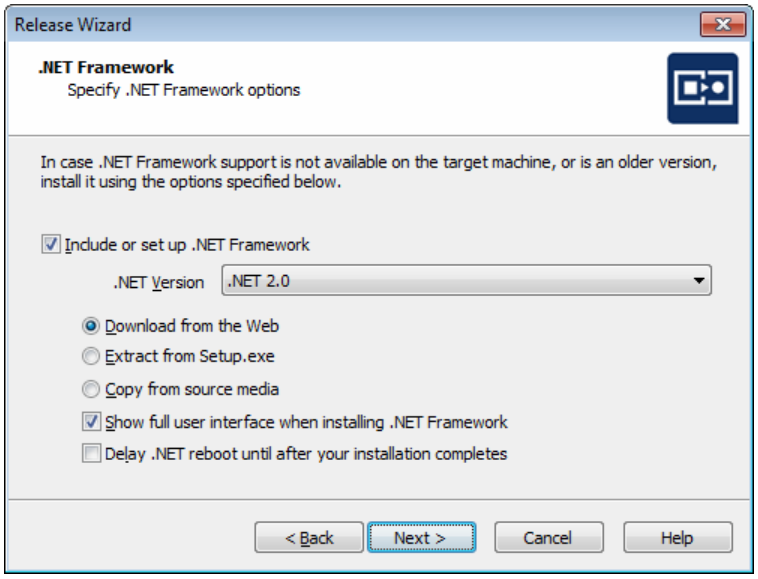

Figure 4-13: Release Wizard: .NET Framework

Note • *The .NET Framework redistributable (`dotnetfx.exe`) adds approximately 20 MB to the install image.*

If you include the .NET Framework version 2.0, the Release Wizard displays the additional panels, but the language options are disabled as the .NET 2.0 Framework includes all of the language support.

Similarly, if you select to include the .NET Framework version 1.1 redistributable, the Release Wizard displays additional panels (not displayed) for .NET run-time options and .NET language pack run-time options, as well as a panel for installing the Visual J# components.

To include a .NET Framework version later than version 2.0, or to include a 64-bit version of the .NET Framework, you can add the corresponding redistributable to your project. Redistributables were discussed in Chapter 2, Creating an Installation Project.

In the **Advanced Settings** panel you specify additional settings for the release, such as the build location and whether to build the release image to use long file names.

For a CD-ROM or DVD-ROM installation, you should select **Generate Autorun.inf**, which enables AutoPlay for the disc.

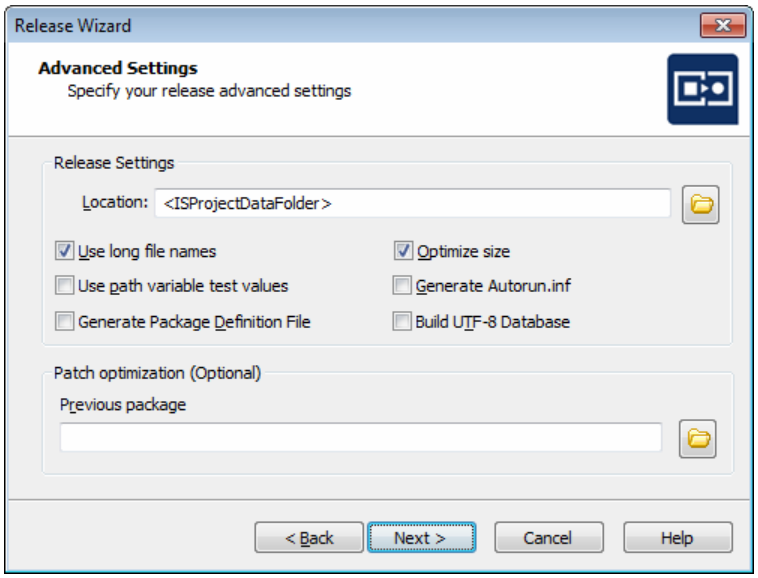

Figure 4-14: Release Wizard: Advanced Settings

The **Optimize size** option improves the compression used when compressing data files into cabinet files (using LZX instead of MSZIP compression), but it can cause the build process to take more time. For quicker, testing-only builds of large releases, you might want to leave **Optimize size** unchecked.

The **Build UTF-8 Database** option generates a Windows Installer database that supports the UTF-8 encoding. The UTF-8 encoding enables the database to support characters from all languages simultaneously. In the event that multiple languages are contained within the database or any UTF-8 transforms, each of the language strings is displayed correctly independent of the current language of the operating system.

The **Previous package** field in the **Patch optimization (Optional)** group lets you specify the MSI database for an earlier version of your product, which ensures an update installer uses the same identifiers in the File, Component, Media, and other tables as a previous release. Doing so ensures that a subsequent patch functions effectively.

The procedures and issues involved in creating update packages are described in Chapter 10, Updates and Patches.

To build your release image after viewing the **Summary** panel (not pictured), click **Finish**, verifying that the **Build the Release** check box is selected.

As a release is building, InstallShield displays progress messages in the **Build** tab of the output window at the bottom of the IDE. Among other things, the entries in the progress messages describe the database tables being built into the MSI database.

The MSI database tables are described in Chapter 8, Conditions and Actions.

To stop a running build, pull down the **Build** menu and select **Stop Build**, press **Ctrl+Break**, or click the **Stop Build** icon in the toolbar.

When the build process is complete, the output window contains a hyperlink to the build log file, in which you can review the messages created by the current build.

If an error occurs, one or more error messages appear in the output window. For example, InstallShield cannot perform a release build if the build folder is open in Windows Explorer or a command prompt. The InstallShield Help Library topic "Troubleshooting Build Errors" lists error codes and troubleshooting tips.

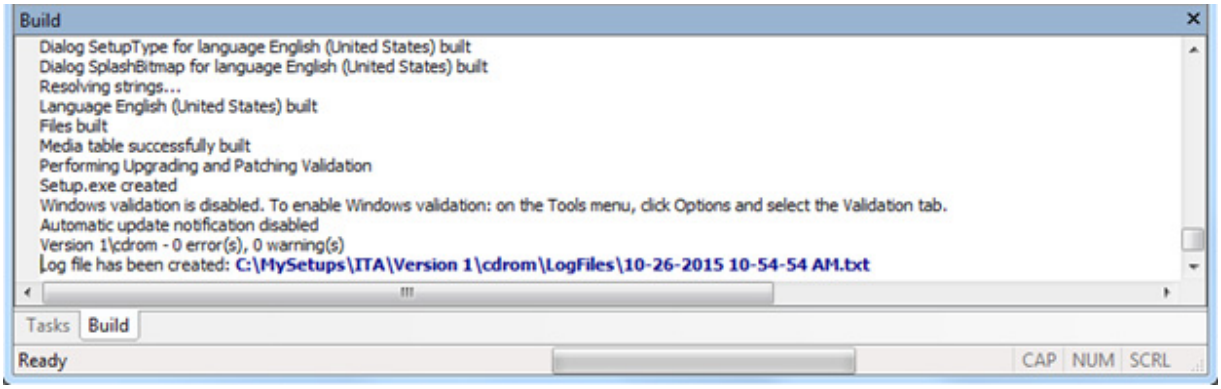

Figure 4-15: Output window

Any build errors or warnings will also appear in the **Tasks** tab of the InstallShield output window. The **Tasks** tab displays a list of hyperlinks to the Flexera Software knowledge base, giving up-to-date information about how to address these errors and warnings.

You can configure InstallShield to abort a build when the first error occurs by pulling down the **Tools** menu, selecting **Options**, and checking **Stop build process when first error is encountered** on the **General** tab.

Files Created by the Release Wizard

By default, your release images are placed in the folder:

*Projects Folder**Project Name**Product Configuration Name**Release Name*\DiskImages\Disk1

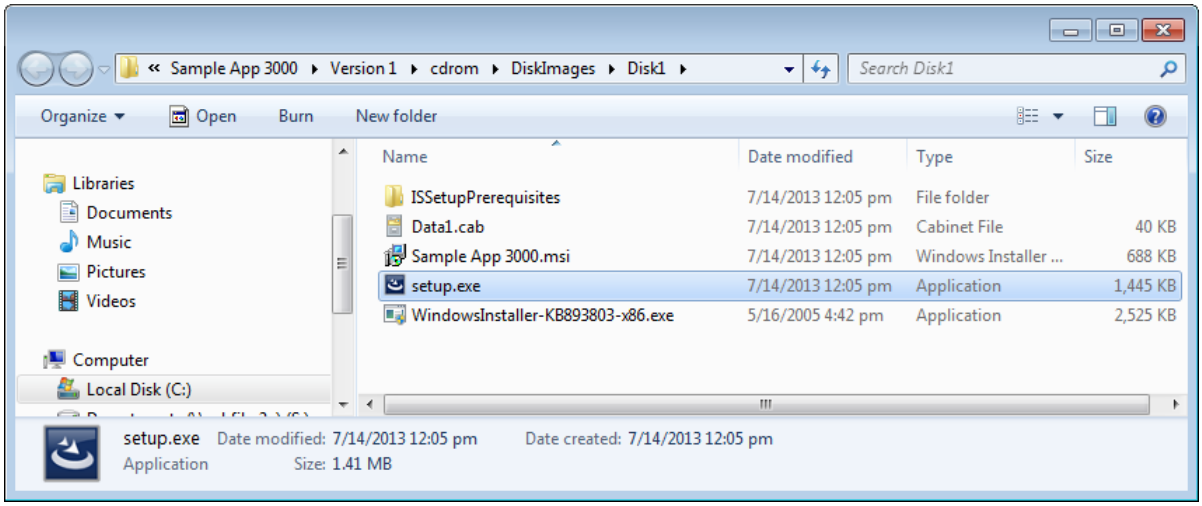

Figure 4-16: Example of files included in a release image

Tip • *You can click the **Open Release Folder** toolbar button to open your release folder in Windows Explorer. However, InstallShield cannot build a release if the release directory is open in Windows Explorer or a **Command Prompt** window, or if an instance of the installation program is running or otherwise locked.*

If your release images span multiple disks, folders called `Disk2`, `Disk3`, and so forth, are created in the same build location.

The files included in a compressed build release are the following:

Table 4-1 • Files Included in a Compressed Build Release

File	Description
`Setup.exe`	Checks if Windows Installer exists on a target machine, and launches the ANSI or Unicode version of the MSI redistributable if necessary. Depending on your settings in the Release Wizard, `Setup.exe` can also launch the .NET Framework and other redistributables and prerequisites. For a multi-language installation program, `Setup.exe` can also display the language-selection dialog box, as described in Chapter 9, User Interface. After it has displayed any necessary dialog boxes and launched associated redistributables, `Setup.exe` launches the MSI database, described below.

Table 4-1 • Files Included in a Compressed Build Release

File	Description
ProductName.msi	The MSI database built by the Release Wizard. By default, the name of the MSI file is based on your ProductName value. You can override the MSI name by selecting a product configuration icon in the **Releases** view and specifying the desired name in the **MSI Package File Name** field.
Data1.cab	Your compressed data files.
WindowsInstaller-KB######-x86.exe	Windows Installer 3.1 redistributables, if selected.

If you choose **Uncompressed and separate from the installation package** in the **Release Configuration** panel of the Release Wizard, uncompressed application files are copied into folders called Program Files, Windows, and so forth, also located in the Disk1 folder. Prerequisites will be found in the ISSetupPrerequisites subfolder.

To minimize the amount of data in the disk images, you can specify URLs from which to download the Windows Installer and .NET Framework redistributables in the **Windows Installer Location** and **.NET Framework** panels of the Release Wizard. The same thing can be done for .NET InstallShield prerequisites.

Note • *When burning a CD-ROM or DVD-ROM, you must make the volume label the same value that appears in the VolumeLabel field of the Media table of your built MSI database. By default, the volume label is DISK1 (for a single-disk installation; subsequent discs must be labeled DISK2, DISK3, etc.). You can modify the volume label to use in the custom spanning options.*

Additional Product Configuration Properties

While the Release Wizard covers the main options for defining a configuration/release image, there are other properties available through the build configuration name and release name. The build configuration properties are shown in the figure below.

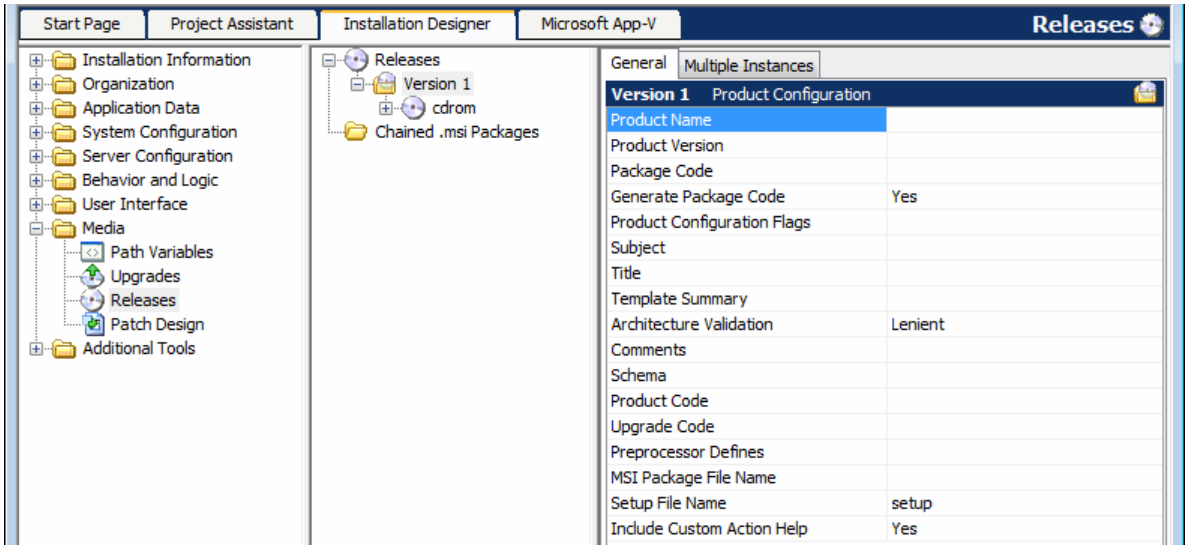

Figure 4-17: Product Configuration properties

The majority of these settings override MSI database values such as **ProductName**, **ProductVersion**, and **UpgradeCode**. You can also specify a different name for the MSI database or setup executable. Other settings override information stored in the Summary Information Stream, such as **PackageCode**, **Title**, **Template Summary**, **Comments**, and **Schema**.

Starting with InstallShield 2013, the **Architecture Validation** setting enables you to create a pure 32-bit or 64-bit installation package, based on the value of the Template Summary setting. If Architecture Validation is set to "Lenient" (the default), you can use a mix of 32-bit and 64-bit files in your installation project. If set to "Strict", InstallShield consults the value of Template Summary, either in the project's Summary Information Stream settings or any override value you provide in the product configuration settings: If Template Summary is set to "Intel", InstallShield validates that only 32-bit files are included in the project, and if Template Summary is set to "x64", InstallShield validates that only 64-bit files are included. At build time, InstallShield reports build warnings and errors if any files do not match the target architecture:

```
ISDEV : warning -7326: Including 32-bit PE file C:\Example\Sample.exe in a strict 64-bit package.
```

When strict validation is used, InstallShield provides 32-bit or 64-bit versions of custom action DLLs, based on your desired architecture.

Additional Release Properties

A number of other build settings are available for each release. If you select a release icon in the **Releases** view, you can view and modify the properties you set in the Release Wizard. In addition, there are some properties that are available only in the property list in the **Releases** view. The release settings are spread across six tabs: **Build**, **Setup.exe**, **Signing**, **.NET/J#**, **Internet**, and **Events**.

Build Tab

The **Build** tab provides access to settings specific to the build image itself. The settings are shown in the picture below.

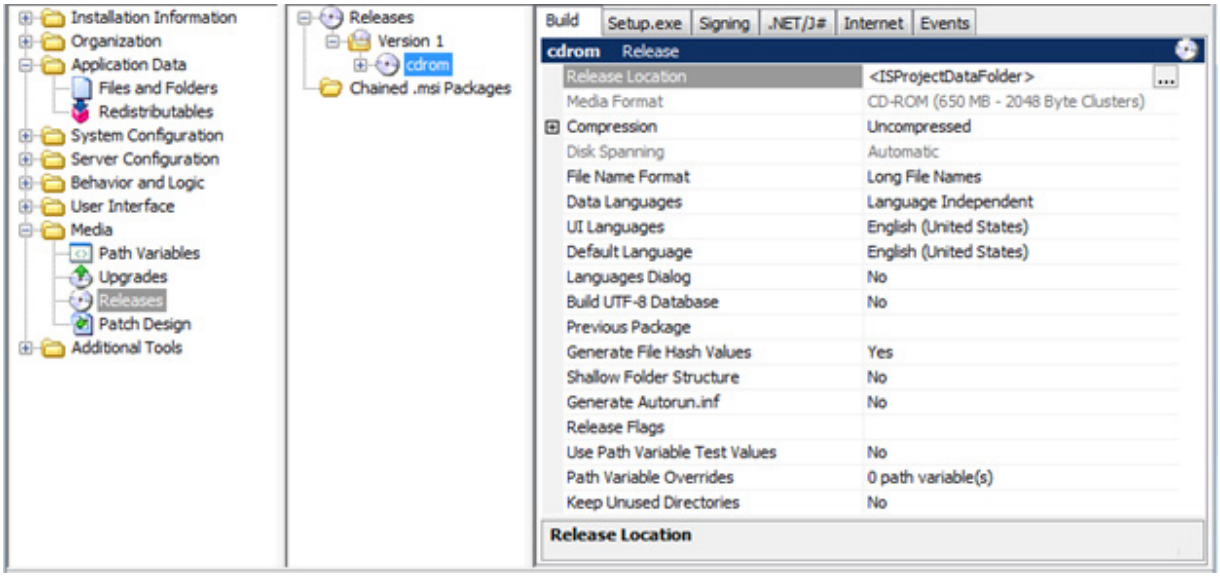

Figure 4-18: Build tab in release properties

Some of the additional build release properties are:

- **Compression**—Specifies whether to compress the product application files.
- **Build UTF-8 Database**—Specifies whether to generate a Windows Installer database (`.msi`) that supports the UTF-8 encoding. The UTF-8 encoding enables the database to support characters from all languages simultaneously. In the event that multiple languages are contained within the database or any applied UTF-8 transforms, each of the language strings is displayed correctly independent of the current language of the operating system.
- **Generate File Hash Values**—Specifies whether to populate the `MsiFileHash` table of your MSI database at build time. Windows Installer uses the contents of the `MsiFileHash` table to determine whether to overwrite an unversioned file that exists on the target system. For details, see the MSI Help Library section "Default File Versioning".

- **Shallow Folder Structure**—Specifies whether your release image files should be built into the *ReleaseName**ProductConfig**DiskImages**DiskN* directory structure. The default value is **No**. If you set this property to **Yes**, and your project uses the Create Project in Project Name subfolder, the disk images will be placed directly in the release's **Release Location** folder. When this setting is set to **Yes**, the build log and report files are still generated. However, instead of being placed in the target folder with the build image, a similarly named target folder with the string `Data` appended is created. This new folder contains the build output data and files. (If you did not select the option "Create project file in 'Project Name' subfolder", the disk images will be placed directly in a subdirectory called [*ProductName*] of the release's Release Location folder.)
- **Path Variable Overrides**—Specify individual overrides to defined path variables. The override occurs at build time for the selected release. You can override any user-defined path variables, environment variables, and registry variables that are configured in the **Path Variables** view. You cannot override predefined path variables.
- **Keep Unused Directories**—Specifies whether you want InstallShield to remove unused directories from the **Directory** table of the `.msi` file when you build this release. Available options are:
  - **No**—This is the default value. If a directory that is listed in the **Directory** column of the **Directory** table is not referenced in any known location in the `.ism` file, InstallShield removes it from the **Directory** table of the `.msi` file that it creates at build time. This occurs after any merge modules are merged, but only directories that are present in the `.msi` file are removed; therefore, if a merge module contains new unused directories in its **Directory** table, these new unused directories are added to the installation.
  - **Yes**—InstallShield does not remove any directories from the **Directory** table of the `.msi` file that it creates at build time.

Note • *Under some conditions, predefined directories cannot be resolved, causing an installation to fail. Removing unused directories from the* ***Directory*** *table enables you to avoid unnecessary failures. Therefore, it is recommended that you select* ***No*** *for this setting.*

Setup.exe Tab

The **Setup.exe** tab contains settings for packages built to include the install executable.

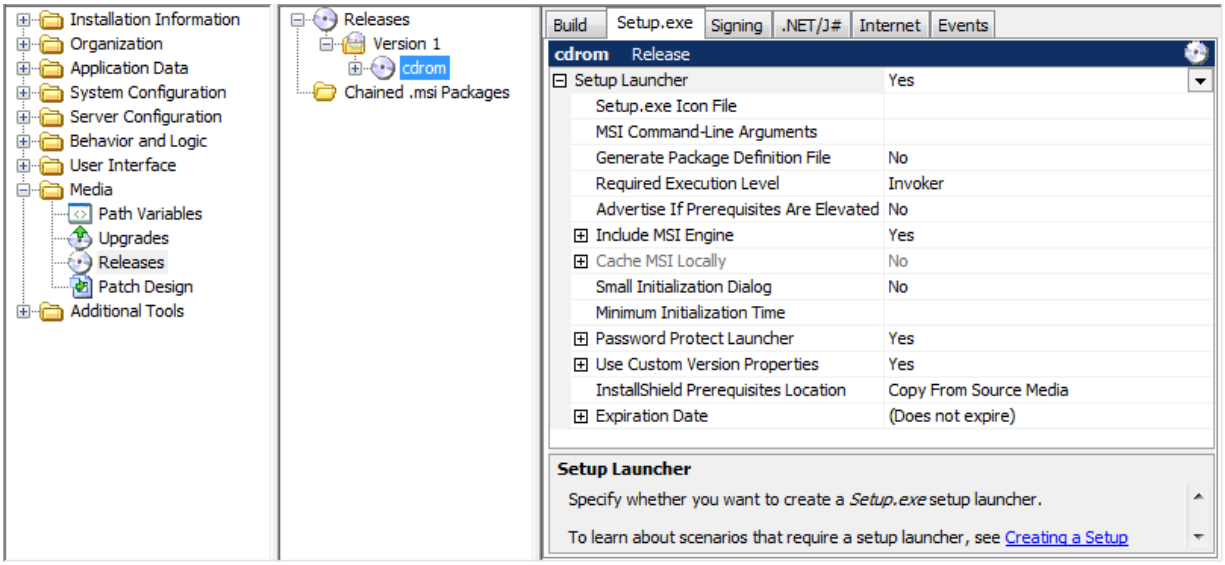

Figure 4-19: Setup.exe tab in release properties

Some of the additional Setup.exe release properties are:

- **Setup.exe Icon File**—Specifies an alternative `Setup.exe` icon. To use your own icon for the `Setup.exe` file, specify the fully qualified name of the file that contains the icon. To specify a file, type an absolute path or a path that is relative to a path variable, or click the ellipsis button (...) to browse to the file from within the **Change Icon** dialog box.
- **MSI Command-Line Arguments**—Specifies command-line arguments and property values to pass to `msiexec.exe` when the user launches `Setup.exe`. `Msiexec.exe` is described in Chapter 5, Deploying an Installation.
- **Required Execution Level**—Defines the minimum level required by your installation's `Setup.exe` file (the setup launcher, any setup prerequisites, and the `.msi` file) to run on Windows Vista and later operating systems. Note that an end user's installation experience is most secure when the installation runs at the lowest permission level it requires.
- **Minimum Initialization Time**—Controls the amount of time (in seconds) that the **Initialization** dialog is displayed. Because the splash screen is displayed at the same time as the Initialization dialog, this setting effectively sets the minimum time the splash screen is displayed. This is especially helpful when initialization is very brief.
- **Expiration Date**—To prevent end users from being able to run `Setup.exe` on or after a certain date, enter the expiration date, or click the arrow in this setting to select the date from a calendar. To remove the expiration date, click the **Delete** button in this setting. If you specify an expiration date in this setting, use the **Expiration Date** setting to specify the message that you want to be displayed at run time if an end user tries to launch `Setup.exe` on or after the expiration date.

Signing Tab

Signing your setup requires purchasing a Digital Signature from a Digital Certificate provider. The **Signing** tab contains settings that control applying certificates to your package.

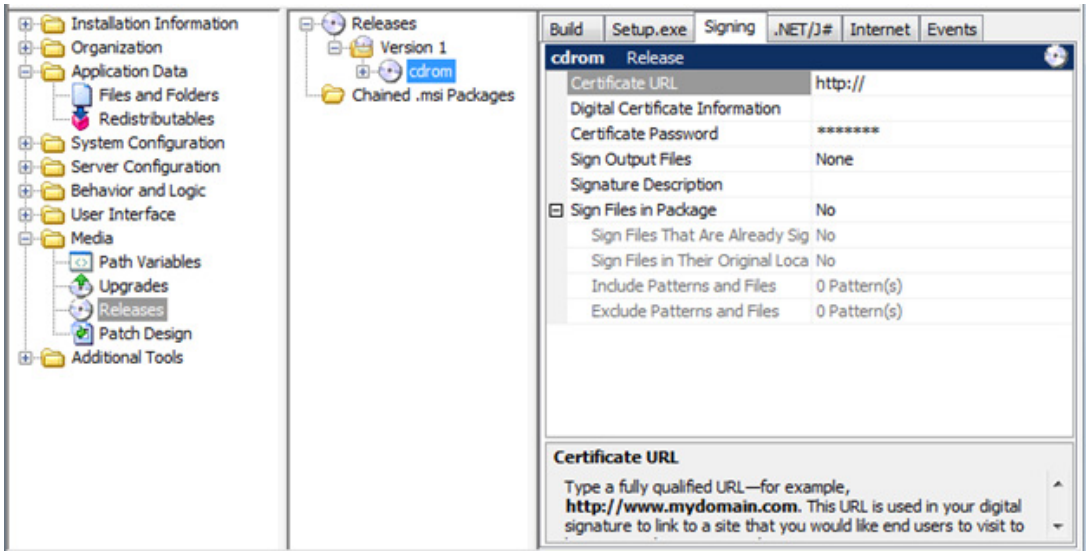

Figure 4-20: Signing tab in release properties

Some of the additional signing release properties are:

- **Sign Output Files**—Specifies which file(s) to sign (`Setup.exe` and/or Windows Installer package).
- **Sign Files in Package**—Specifies that files within the install image are to be signed. This setting uses the Include/Exclude patterns and files settings.
  - **Sign Files That Are Already Signed**—Specifies that files that have been previously signed should be re-signed using the provided certificate. This setting uses the Include/Exclude Patterns and Files settings.

***Note** • When signing application files, a temporary copy of the file is made. The temporary file is then signed and added to the media image. The original source file is never modified or signed.*

Through the use of the **Signing** tab, you can select which files should be signed in your MSI package. The signature can also be applied to the MSI itself allowing for automatic support of elevation free patching.

***Note** • Starting with InstallShield 2015, SHA-256 certificates are used for signing installations and their application files. These certificates can either be specified by (`.pfx`) filename or you can reference a certificate in a certificate store. Previous versions of InstallShield used a SHA-1 hash. Beginning January 1, 2016, files signed using SHA-1 hashing will not be recognized as validly signed.*

On systems running Windows Vista or later, signing your setups has a direct impact on how your setup will be presented to the end user. Without a signature, the UAC dialog specifies that "an unidentified application" requires elevated permissions. With a digital signature, the UAC dialog displays the actual product and company name.

.NET/J# Tab

While the Release Wizard provides a **.NET Framework** panel, there are a number of additional options that can be configured. The .NET/J# tab contains settings modify the redistribution of .NET and J#.

As with the Release Wizard, this tab controls installation of early versions of the .NET redistributable (through .NET 2.0). For more recent versions, use the **Redistributables** view.

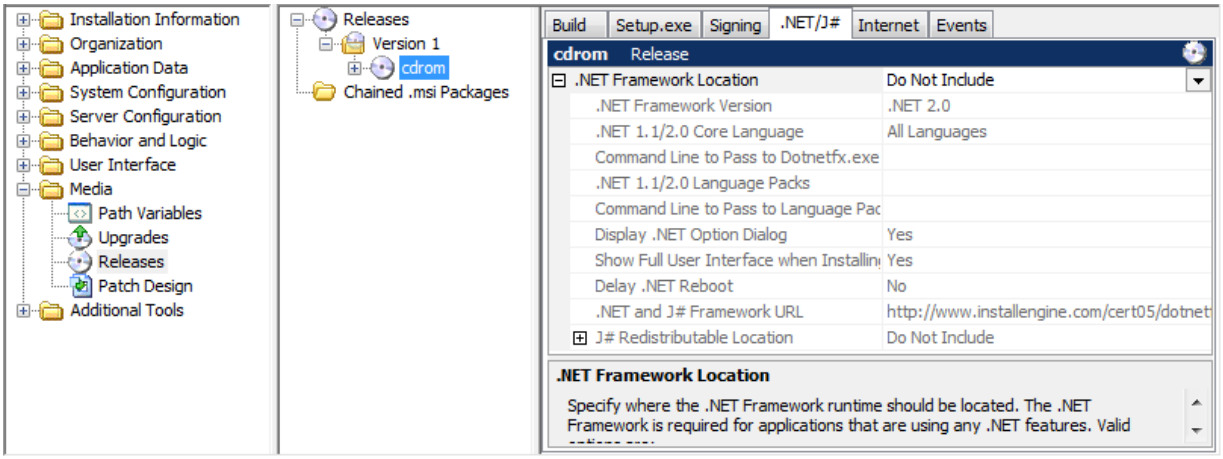

Figure 4-21: .NET/J# tab in release properties

Some of the additional .NET/J# release properties are:

- **Command Line to Pass to Dotnetfx.exe**—Enter any command line values that should be passed to the .NET redistributable. This property applies only if .NET 1.1 is selected for the **.NET Framework Version** property.
- **Display .NET Option Dialog**—Specifies whether the end users have the option to install the .NET redistributable.
- **.NET and J# Framework URL**—Specifies the URL from which to download the .NET/J# redistributable. This property is required only if the **.NET Framework Location** or **J# Redistributable Location** property (if included) for the current release is set to **Download from the Web**.
- **Command Line to pass to the J# Redistributable**—Enter any command line values that should be passed to the J# redistributable.

Internet Tab

The **Internet** tab contains settings that control distribution of an MSI package distributed over the Internet. This tab uses the information gathered from the **Media Type** panel in the Release Wizard. When you select the Web media type from the **Media Type** panel of the Release Wizard, the following options are available: **One Executable**, **Downloader**, and **Install from the Web**. Any of these can also be deployed as a One-Click Install. The **Internet** tab enables you to change and configure the Web media type options.

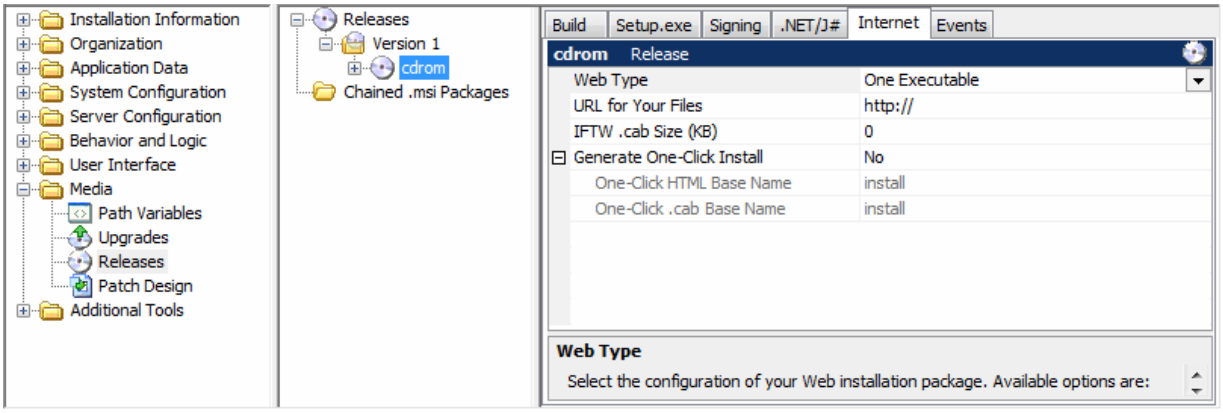

Figure 4-22: Internet tab in release properties

Some of the additional Internet release properties are:

- **IFTW .cab Size (KB)**—Controls the cabinet size for the Install for the Web image. Setting this property to **0** generates a cabinet file for each component.
- **Generate One-Click Install**—Specifies file names for the One-Click Install.

Events Tab

The **Events** tab provides a means of launching events and distributing the image after the build is finished. There are three events available: **Prebuild**, **Precompression**, and **Postbuild**. The distribution options are to copy to a folder,, FTP site, or virtual machine.

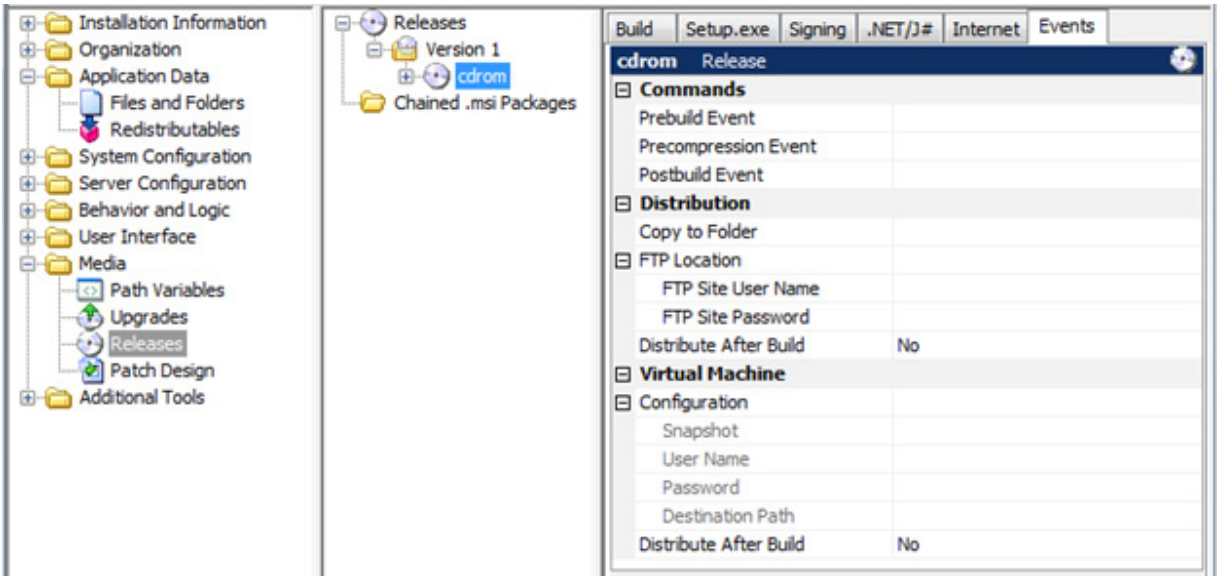

Figure 4-23: Events tab in release properties

Unlike the other tabs, none of the commands or properties in the **Events** tab are available through the Release Wizard.

- **Commands**—Specifies commands that you want to run at various stages of the build process. To specify more than one command, click the ellipsis button (`...`) associated with the event. In the subsequent **Event** dialog box, enter each command on a separate line. InstallShield runs each command in the order that they are listed. When you are entering a command, you can use any path variables and environment variables that are defined in your project, instead of using a hard-coded path. Commands are only available in the Premier edition of InstallShield.
  - **Prebuild Event**—Specify the command that you want to be run before InstallShield starts building the release. This event runs after InstallShield creates the release folder and log file, but before InstallShield starts building the release
  - **Precompression Event**—Specify the command that you want to be run after InstallShield has built the `.msi` package and the `.cab` files (if your product's data files are to be stored in `.cab` files). Note that this event occurs after `.cab` files are streamed into the .msi package, but before the `.msi` package has been digitally signed and streamed into the `Setup.exe` file.
  - **Postbuild Event**—Specify the command that you want to be run after InstallShield has built and signed the release.
- **Distribution**—Specifies a folder or an FTP site to copy your release image at the end of the build or on demand.
  - **Copy to Folder**—Defines a folder location (fixed, mapped, or URL) to copy the media image. This can be done as part of the build, or at any time by right-clicking the build release name and selecting **Distribute**.

- **FTP Location**—Defines an FTP location to copy the media image. This can be done as part of the build, or at any time by right-clicking the build release name and selecting **Distribute**.
  - **Distribute after build**—Specifies whether the **Copy to Folder** or the **FTP Location** should be used after performing a build. If this property is set to **No**, the distribution can still be performed by right-clicking on the build release name and selecting **Distribute**. If you specify a folder location and an FTP location, InstallShield copies the release to only the FTP location.
- **Virtual Machine**—Defines a virtual machine that will be provisioned at the end of the build or on demand. The virtual machine can be on a Microsoft Hyper-V Server, a VMware ESX or ESXi Server, or a VMware Workstation To distribute the release to the VM at any time, right-click the release in the Releases view and then click Distribute to VM. Virtual machine distribution is only available in the Premier Edition of InstallShield.
- **Configuration**—Specifies the snapshot name, user name, password, and destination path for the release image on the virtual machine.
  - **Distribute after build**—Specifies whether the release is to be deployed to the specified VM. If set to **Yes**, when the build completes, InstallShield will revert the VM to named snapshot, power on the machine, logon, and copy the media image to that folder. If this property is set to **No**, the distribution can still be performed by right-clicking on the build release name and selecting **Distribute to VM**.

Rebuilding Releases

To rebuild a release after specifying its properties, click the **Build** toolbar button, pull down the **Build** menu and select **Build**, or press **F7**. You can also rebuild a release from the **Releases** view of the IDE by right-clicking a release name and selecting **Build**. In addition, you can rebuild some or all of the releases under a product configuration by right-clicking the product configuration icon and selecting **Batch Build**. You can also clone, or duplicate, an existing release by right-clicking its icon and selecting **Clone**.

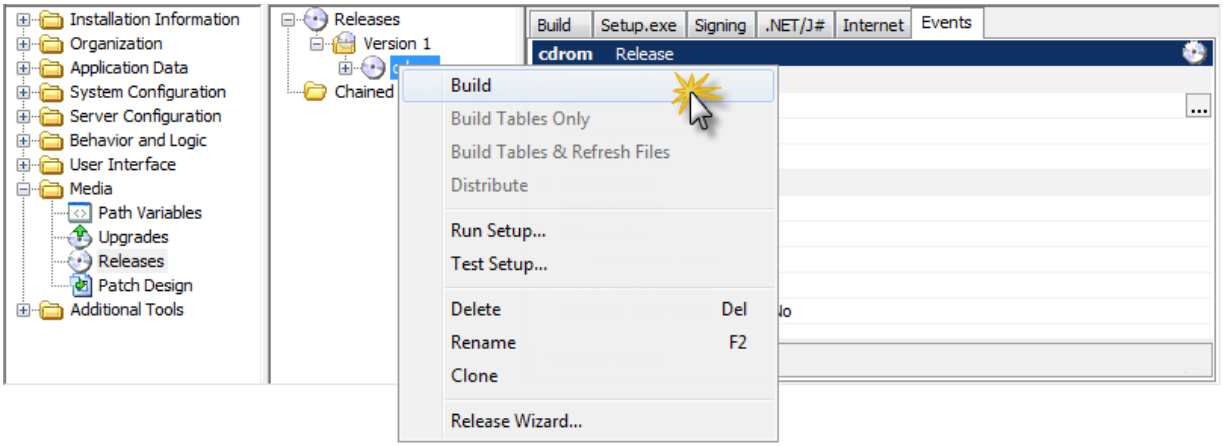

Figure 4-24: Building a release in the Releases view

For testing purposes, you can build a release without updating any files by pulling down the **Build** menu and selecting **Build Tables Only**, or by right-clicking a release icon in the **Releases** view and selecting **Build Tables Only**. You can also rebuild your MSI database and update only those files that have changed since the previous build by selecting **Build Tables & Refresh Files** from the **Build** menu.

Web Builds

InstallShield also supports various Web media types. For any Web build type, you begin by selecting **Web** in the **Media Type** list on the **Media Type** panel of the Release Wizard. Next, select the desired Web type from the **Web Type** panel of the Release Wizard.

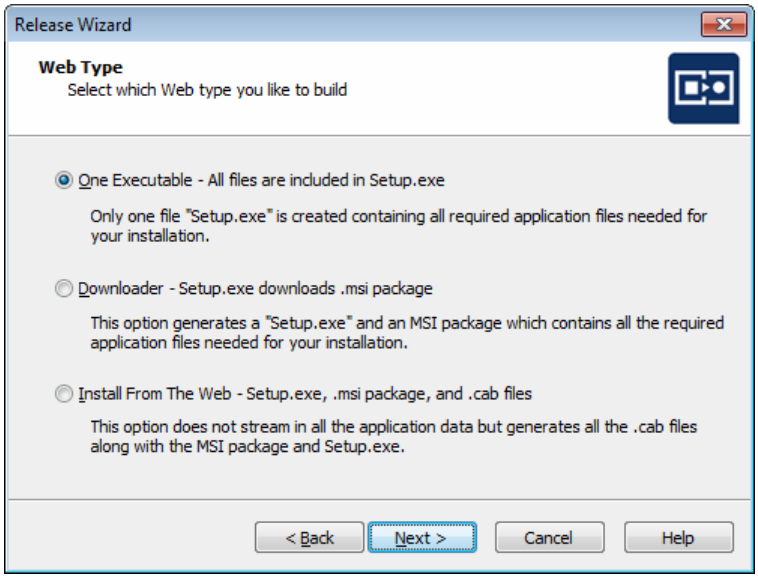

Figure 4-25: Release Wizard: Web Type

The Web types you can use are:

- **One Executable**—This media type creates a single `Setup.exe` that the user can download from a Web site or FTP site. The advantage to a **One Executable** build is that the installation program is self-contained, and not associated with a specific URL.
- **Downloader**—This media type creates the `Setup.exe` installation launcher and a separate MSI database. The user downloads and launches `Setup.exe`, which in turn downloads and runs the MSI database from a URL you specify. You can also specify whether the Downloader should use external `.cab` files, and if so, whether to create `.cab` files based on features, components, or file size.

  For this media type, you can optionally specify a directory where your data files will be cached on the target system. Selecting this option is strongly recommended for a One Executable build: when the user launches `Setup.exe` for a One Executable build, the MSI database and data files are expanded into a temporary directory, the installation runs from the temporary directory, and then the temporary directory is deleted.

  When the user runs **Repair** for an application, the Windows Installer service will prompt for the original source location, which no longer exists. At this point, the user must again download the executable image and decompress the executable to expose the MSI database and data files. Specifying a cache path in the **Local Machine** panel avoids this issue by caching the data files on the target system.

Note • *If you use the option to cache files on the local system, the cached files are not removed when the user uninstalls your product. They must be manually removed by the user when the image is no longer needed.*

- **Install from the Web**—This media type creates the `Setup.exe` installation launcher, the MSI database, and one or more external cabinet (CAB) files containing your application data. The user launches `Setup.exe`, which downloads the MSI package from the URL you specify, and then downloads only the cabinet files required by the user's setup-type and feature selections. For this media type, you specify whether to create a single cabinet file for each component or for each feature, or else to create cabinet files of a specific size.

Note • *The definition of the Media table of an MSI database requires the cabinet files to be created with short filenames. While handled automatically by InstallShield, the cabinet filenames must conform to the short filename convention—eight characters followed by a dot and a three-character file extension.*

One-Click Installs

For any Web build type, you can create a One-Click Install, which uses an HTML page as the initial user interface of the installation program.

Figure 4-26: Initial user interface for One-Click Install installation

To build a One-Click Install, select **Generate a One-Click Install** in the **One-Click Install** panel of the Release Wizard.

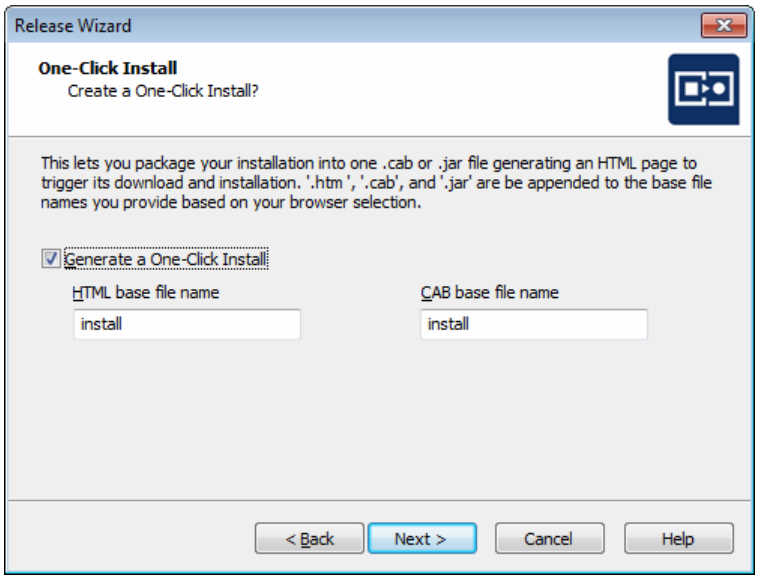

Figure 4-27: Release Wizard: One-Click Install

In this panel, specify the name of the HTML file to generate (the `.htm` extension will be added to the name you specify), and the name of the cabinet file to generate. The `.cab` extension will be added to the name you enter in the **CAB base file name** field.

Building Releases from the Command Line

To build releases from a batch file or command prompt, you can use `IsCmdBld`, located in the `System` subfolder of the InstallShield distribution folder. While `IsCmdBld` can be used to create new product configurations and releases, it is most commonly used to rebuild product configurations and releases that you previously configured with the Release Wizard.

To be able to run `IsCmdBld` from any directory, you must add the InstallShield System subfolder to the `PATH` environment variable or specify the absolute path to `IsCmdBld` in your batch file or other build script.

When building a release with `IsCmdBld`, the following command-line arguments are required or recommended:

- The `-p` *`project_location`* switch specifies the full path to the InstallShield project file (`.ism` file) that you want to build. A typical value is `C:\MySetups\ITA.ism`. The project location must be an absolute path; you can use a UNC path, if desired.
- The `-r` *`release_name`* switch specifies which release (`FirstBuild`, for example) you want to build. You can build a new release by specifying a name not used for an existing release. (The `-r` switch is optional.) Release names are case sensitive.

- The `-a` *`product_config`* switch specifies the product configuration to use. (The `-a` switch is optional.) Product configuration names are case sensitive.

For example, to rebuild the ITA release called `dvd` in the product configuration `Version 1`, you can use the following command line.

```
iscmdbld -p "C:\MySetups\ITA.ism" -r dvd -a "Version 1"
```

Status, warning, and error messages for the build process are displayed in the command-prompt window. These are the same messages displayed in the output window when you build from the IDE. You can suppress these status messages by adding the `-s` argument to the `IsCmdBld` command.

```
InstallShield (R)
Release Builder
Copyright (c) 2015 Flexera Software LLC.

All rights reserved.

Build started at Sep 10 2015 07:02 AM
Building Release: dvd
    [...lines omitted...]
Resolving strings...
Language English (United States) built
Building CAB files...
Data1.cab built
Files built
Media table successfully built
Performing Upgrading and Patching Validation
Setup.exe created
Version 1\dvd build completed with 0 errors, 0 warnings
Log file has been created: <file: C:\MySetups\ITA\...>

Build finished at Sep 10 2015 07:04 AM
```

If a build is successful, `IsCmdBld` sets `ERRORLEVEL` to 0, and if the build fails, `ERRORLEVEL` is set to 1. To stop a command-line build if an error occurs, pass `IsCmdBld` the `-x` switch. To treat build warnings as errors, use the `-w` switch.

Other command-line arguments available with `IsCmdBld` are described in the InstallShield Help Library topic "Building a Setup from the Command Line", or by launching `IsCmdBld` with no arguments at a command prompt. You can also build a release using the **Build** method of the InstallShield Automation interface, described in the following section.

Automation

Throughout this course, you have seen how to modify project properties using the different views and wizards of the InstallShield environment. Another way to read and modify the properties of your installation project is using the Automation interface.

The Automation interface enables you to change properties of your project, or build your project, from a VBScript file or other language or process that can call COM methods, without needing the InstallShield environment to be open.

Project Properties

The ProgID for the InstallShield 2015 Automation interface is `ISWiAuto22.ISWiProject`. (Earlier product versions used different `ProgID` values: Consult your product documentation if you have an earlier version of InstallShield.) The general framework for a VBScript file that queries or modifies a project file is the following.

```
Set oProject = CreateObject("ISWiAuto22.ISWiProject")
oProject.OpenProject "C:\MySetups\ITA.ism"
'  perform changes here
oProject.SaveProject
oProject.CloseProject
Set oProject = Nothing
```

For example, the following script sets the value of the `ProductVersion` property of your project.

```
Set oProject = CreateObject("ISWiAuto22.ISWiProject")
oProject.OpenProject "C:\MySetups\ITA.ism"
oProject.ProductVersion = "1.2.3"
oProject.SaveProject
oProject.CloseProject
Set oProject = Nothing
```

Using the same technique, you can read or modify the following global project settings, which are exposed as read-write properties of the `ISWiProject` object:

- INSTALLDIR
- ProductName
- ProductCode
- ProductVersion
- PackageCode
- UpgradeCode
- ActiveLanguage

The InstallShield Automation interface also provides a `GenerateGUID` method, which generates a new, random GUID for use in project properties such as `PackageCode` and `ProductCode`.

You can also read and modify your project's Summary Information Stream properties using the `ISWiSISProperties` collection.

Features, Components, and File Links

An ISWiProject object contains collection objects that represent all of the features, components, and files contained in your project. Each of the collections—ISWiFeatures, ISWiComponents, and ISWiFiles—contains Count and Item properties, which enable you to enumerate all of the individual items in the collection.

The ISWiFeatures collection lists only the project's top-level features. To enumerate a feature's subfeatures, you can use the Features property for each individual ISWiFeature object. For example, the following VBScript recursively enumerates all the features contained in a project.

```
Set oProject = CreateObject("ISWiAuto22.ISWiProject")
' open project as read-only
oProject.OpenProject "C:\MySetups\ITA.ism", True
' loop over all top-level features
For Each top_feature in oProject.ISWiFeatures
    WScript.Echo top_feature.Name
    EnumFeaturesAndSubFeatures(top_feature)
Next ' top_feature
oProject.CloseProject
Set oProject = Nothing

' recursively list a feature's subfeatures
Sub EnumFeaturesAndSubFeatures(m_Feature)

For Each feature In m_Feature.Features
    WScript.Echo feature.Name
    EnumFeaturesAndSubFeatures(feature)
Next ' feature
End Sub
```

Furthermore, you can add features and subfeatures to your project using the AddFeature method; add components to a feature using the AddComponent method; and add file links to a component using the AddFile method.

Building from the Automation Interface

You can also build a release using the Build method of the InstallShield Automation interface. A sample VBS file (build.vbs) that builds a release using the Automation interface might appear as follows:

```
Set oISM = CreateObject("1ISWiAuto22.ISWiProject")
' open the project as read-only
oISM.OpenProject "C:\MySetups\ITA.ism", True
' build the named product configuration and release
oISM.ISWiProductConfigs("Version 1").ISWiReleases("dvd").Build( )
' close the project
oISM.CloseProject
Set oISM = Nothing
```

After creating the VBS file, you can double-click it (or run the command cscript /nologo build.vbs in a batch file) to build the release. Note that on a 64-bit system, it may be necessary to launch the script using C:\Windows\SysWOW64\cscript.exe instead of double-clicking the script. After you build a release with the Build method, the release object's BuildErrorCount and BuildWarningCount properties will contain the number of build errors and warnings generated.

There are also `BuildTablesOnly` and `BuildTablesRefreshFiles` methods that correspond to the **Build Tables Only** and **Build Tables & Refresh Files** commands under the **Build** menu.

Similarly, you can create new product configurations (using the `AddProductConfig` method) and releases (using the `AddRelease` method) using the Automation interface, specifying the properties of each using the `ISWiProductConfig` or `ISWiRelease` object you create. For details, see the InstallShield Help Library topic "ISWiProject Object".

Advanced Note • *InstallShield project files (ISM files) that are stored in binary format are simply renamed MSI files. Therefore, to make project changes not exposed by the InstallShield Automation interface, you can use the MSI Automation interface to modify a binary ISM file. The InstallShield Automation interface is supported for InstallShield projects stored in XML format, but the Windows Installer Automation interface is not. For more information about MSI Automation, see the Windows Installer help library.*

Standalone Build System

InstallShield provides a standalone build system, which enables you to build projects on a system without having to install all of InstallShield. (The standalone build system is included with the Professional and Premier editions of InstallShield 2012 and later, and additional licenses are available for purchase.) An installer for the standalone build tool is available in the `Standalone` folder of your InstallShield installation media. Refer to the InstallShield Help Library topic "Installing the Standalone Build to a Build Machine" for information about obtaining the standalone build installer if you did not obtain InstallShield on physical media.

Note • *The standalone build system can be installed only on a system running Windows 2000 or later.*

At the simplest level, the standalone build system does not require any changes to be made to the build system's registry. The standalone command-line build tool is the same `IsCmdBld.exe` utility described in the previous section. The following additional command-line switches are useful when performing standalone builds:

- The `-o` switch should be followed by a comma-separated list of directories in which merge modules can be found. By default, the standalone build tool does not include any merge modules; you must manually copy any required merge modules to the build system.
- The optional `-t` switch should be followed by the path to the Microsoft .NET Framework redistributable. This switch is required only if your setup program needs to install the .NET Framework on a target system. If you use the `-t` switch, you can also use the `-j` switch, followed by the minimum version of the .NET Framework that you want to install on the target system.
- The optional `-g` switch should be followed by the minimum version of the MSI redistributable that should be installed on the target system.

The standalone build system also supports the Automation interface. The `ProgID` for the standalone build system is the same as the full build system, `ISWiAuto21.ISWiProject`. To use the standalone build system's Automation interface, however, you must either run the Standalone Build tool installation program, or manually register various InstallShield libraries as described in the help topic "Installing the Standalone Build on a Build Machine".

Note • *In very old InstallShield versions, the standalone build executable file was named `IsSaBld.exe`. Moreover, the standalone build system used a different Automation `ProgID`, such as `SAAuto14.ISWiProject` for the InstallShield 2008 standalone build system.*

The InstallShield help library describes Automation properties that are useful for standalone builds, such as `MergeModuleSearchPath` and `DotNetFrameworkPath`.

MSBuild

A feature of .NET 2.0 and later is `MSBuild`, which enables developers to build development projects without having the entire Visual Studio system installed. With InstallShield, you can use `MSBuild` in conjunction with the Standalone Build system to build your development projects and installation projects with the same build script.

If you use the InstallShield environment integrated with the Visual Studio environment, you will find an `.isproj` file along with your `.ism` project file. The `.isproj` file is a build script suitable as input to `MSBuild`. The `.isproj` file is analogous to, for example, a Visual Studio `.csproj` C# project file, which can also be used as input to `MSBuild`.

`MSBuild` functionality can be extended using tasks, and the InstallShield and Standalone Build installations install the task that defines how MSBuild should interpret the elements inside your `.isproj` file. The task DLL should be located in the same directory as `msbuild.exe`, typically `C:\Windows\Microsoft.NET\Framework\`*`version`*`\`.

To build your Visual Studio project and your InstallShield project, you can simply execute `msbuild.exe`, passing it the path to your solution (`.sln`) file and any required arguments:

```
msbuild C:\Projects\SampleApp3000.sln /property:Configuration=Release
```

For information about the parameters of the InstallShield task for `MSBuild`, refer to the InstallShield Help Library topic “Microsoft Build Engine (MSBuild).”

Source Control

You can integrate your project files with any source control system that uses the Microsoft Source Code Control interface. If your development system has a compatible source control system installed, you can launch and configure the source control system by pulling down the **Project** menu and selecting the desired operation from the **Source Control** submenu.

The InstallShield Help Library topic "Using Source Code Control" describes the steps to perform to set up your source control system for integration with InstallShield. For example, the **Source Control** submenu of the **Project** menu contains commands for adding and unlinking your project from source control, and for checking your project in and out of your source control system (as well as getting the latest version of the project without checking it out).

You can also configure how InstallShield interacts with your source control system by pulling down the **Tools** menu, selecting **Options**, and selecting the **Source Control** tab. The settings in the **Source Control** tab enable you to automatically add new projects to your source control system, get the latest project version when you open a project, and check out the project when you make changes to it.

Support Files

The **Support Files** view can be used to add files that are available to the installer on the target system for use only during the installation process. These files are copied to the target system by the first action to be run by the MSI and removed by the final action, meaning they can be accessed at all times during the installation. This can be useful for a chained MSI.

At runtime, these files will be typically extracted to the `%TEMP%\[ProductCode]` subfolder, and although this folder will vary at times, the `[SUPPORTDIR]` property can be used to reference the folder.

For multi-language installations, localized files and folders can be stored under the relevant language node, or files for all languages under the **Language Independent** section.

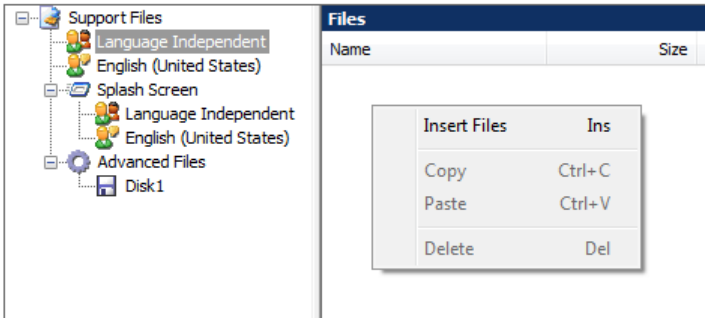

Figure 4-28: Support Files view

Advanced Note • *Unlike other directory properties, The SUPPORTDIR property does not contain a trailing backslash, so remember to add a backslash in any file path using the property, such as:*

```
[SUPPORTDIR]\ita.txt
```

In InstallShield 2014, functionality was added to allow subdirectories to be created in the Support Files folder. This can be done by right-clicking on the root language entries and selecting **New Folder**.

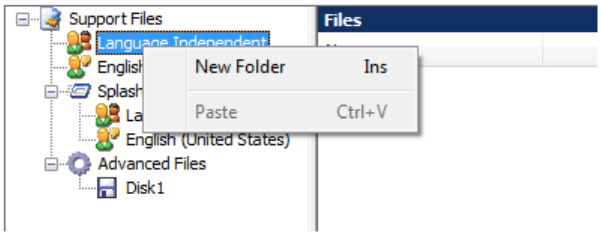

Figure 4-29: Creating folders in Support Files

Files can be added by selecting the relevant folder or language and then right-clicking in the **Files** pane on the right and selecting **Insert Files**.

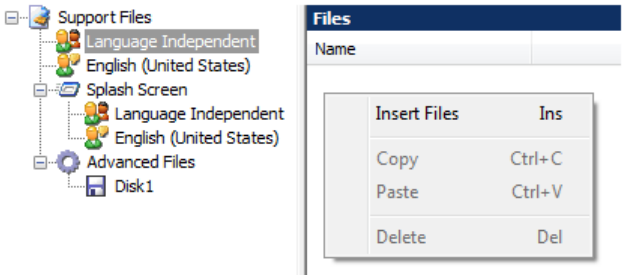

Figure 4-30: Inserting files to Support Files

The **Splash Screen** pane section allows a bitmap to be added to the installation and displayed to the user at run time during the initialization of the installer. As with the support files, localized files can be added by selecting the relevant language node, right-clicking in the **Files** pane on the right and clicking on **Insert Files** in the context menu. These files will be removed from the target at the end of the installation.

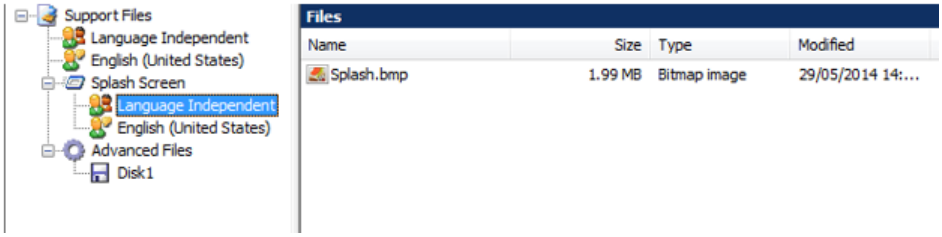

Figure 4-31: Adding a Splash Screen

The final option on the **Support Files** view, **Advanced Files**, allows you to specify files or a folder tree that will be copied, uncompressed, to the installation media at build time.

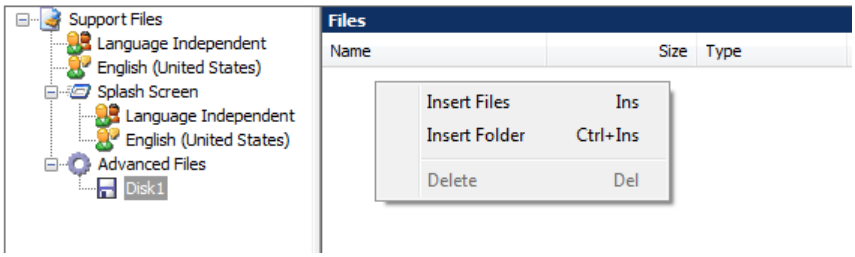

Figure 4-32: Advanced Files

Unlike support files, the files and folders are not streamed into the MSI and will—no matter what build type is selected—show up in the root folder of the built disk image. If required, the files can be accessed by custom actions and other functionality in the MSI using the standard Windows Installer `[SourceDir]` directory property.

Advanced Note • *The **Advanced Files** functionality should not be used with the compressed Network media format, as the files will not be merged into the desired single `Setup.exe` executable, but rather will be placed next to the `Setup.exe` file in the build folder. In this case, it is preferable to use **Support Files** rather than **Advanced Files**, as **Support Files** will be streamed into the MSI and, ultimately, the `Setup.exe`.*