

This partial chapter is an excerpt from the training manual, “Learning InstallShield 11.5® Express Edition.” For information about purchasing this training manual and to learn about other Macrovision training materials, please visit <http://www.macrovision.com/services/education/>.

5

CONFIGURING THE TARGET SYSTEM

In addition to installing files, most installation programs need to perform additional changes to the configuration of the target system. In addition to copying application files, an installation commonly makes other changes to the target system, including adding information to the registry, registering file extensions, and making changes to INI files.

In this chapter, you will learn how to:

- Create and modify shortcuts on the target system.
- Modify the target system’s registry.
- Create and modify INI files.
- Create file associations for your application.
- Create and modify environment variables.

Working with Shortcuts

In Chapter 2, you saw how the Application Shortcuts page of the Project Assistant enables you to create shortcuts to executables and other files contained in your installation project. In this section, you will learn about additional properties used by shortcuts in Windows Installer installation programs.

Windows Installer supports two types of shortcuts: advertised shortcuts and non-advertised shortcuts. Advertised shortcuts are created on the target system when the product (or corresponding feature) is advertised on a target system. (Recall that an advertised product or feature is not installed on the target system until the end user requests it.) A non-advertised shortcut can launch any executable or file, whether it is being installed by the current installation or already exists on the target system.

LOGO

The “Designed for Microsoft Windows” Logo guidelines require you not to create shortcuts to Readme or help files, or to your application’s uninstaller. If you require a shortcut to your application’s uninstaller, you can use the Application Shortcuts page of the Project Assistant, as described in Chapter 2.

You can specify the shortcuts and program folders that you want to create on a target system using the Shortcuts/Folders view, located in the “Configure the Target System” section of the View List.

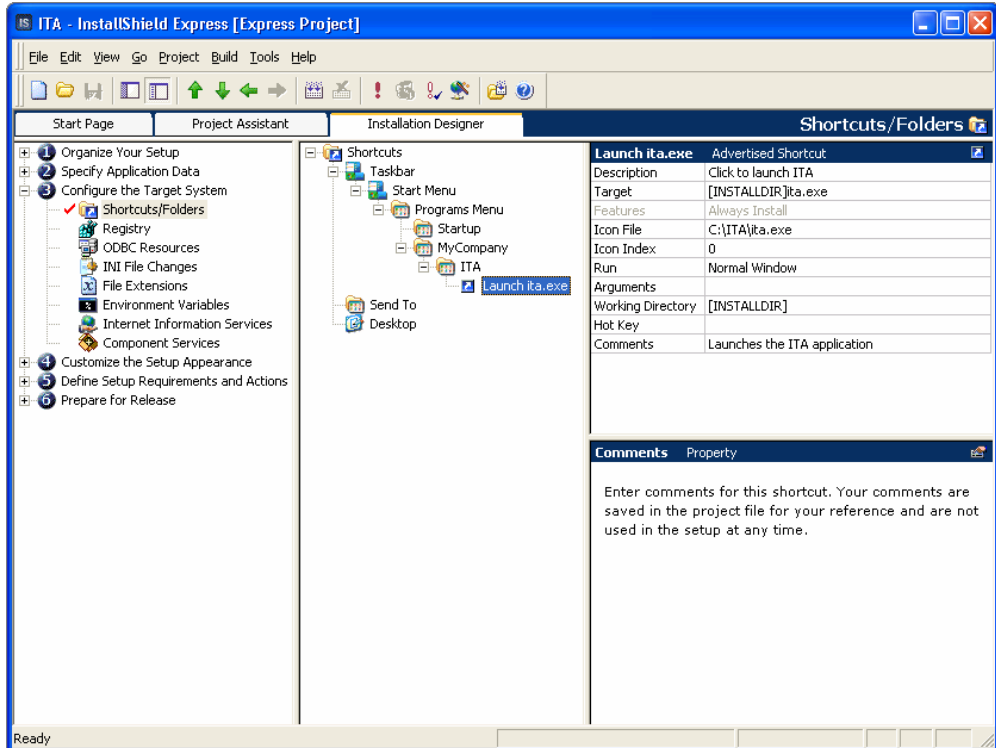


Figure 5-1: The Shortcuts/Folders view in the Installation Designer.

TASK

To create a shortcut:

1. Right-click the icon representing the folder in which you want to create the shortcut—such as Programs Menu—and select “New Advertised Shortcut” or “New Shortcut to Preexisting File,” renaming the shortcut icon as desired. (If you right-click a program folder and select New Shortcut, the Browse for Shortcut Target dialog box is displayed to prompt for a file inside the installation project to use as the shortcut’s target.)
2. Define the shortcut settings (described in the following section).

You can change the location of a shortcut by dragging its icon to a new folder. For example, shortcuts you created in the Project Assistant will be placed in subfolders of the Programs Menu folder named after your company and product names. To indicate that the shortcut should be created at the root of the Programs Menu folder, you can drag the shortcut icon to the Programs Menu icon, and then delete the now-empty subdirectories.

The name you provide your shortcut in the Shortcuts tree is the name that is used when the shortcut is installed on the target system.

Specifying Shortcut Settings

The settings you specify for a shortcut are the following.

- The **Description** value appears in the shortcut's tool tip and property sheet on Windows 2000 and Windows XP.
- The **Target** setting specifies the full path to the file to launch when a user launches the shortcut. For shortcuts to pre-existing files, the path points to where the file is located on the target system. For standard shortcuts and advertised shortcuts, the path points to where the file is located on the development system.
- The **Feature** setting shows the feature with which the shortcut is associated. To ensure that your shortcut is installed along with its target file (if the file is included with the installation), you need to include the shortcut in the same feature as the target file.
- The **Icon File** and **Icon Index** settings specify the icon to display for the shortcut. When you click the browse button next to the Icon File property, the Change Icon dialog box lets you browse for the desired icon.
- The **Run** setting specifies the style of window (normal, maximized, or minimized) the target file should use.
- The **Arguments** setting specifies any command-line arguments to be passed to the application when it is launched with the current shortcut.
- The **Working Directory** setting specifies the working directory used by the shortcut target application (displayed as the "Start in" directory in the shortcut's property sheet). The working directory is the default directory displayed in standard file-opening and file-saving dialog boxes, as well as the current directory used by the application. A typical value is [INSTALLDIR]Files.
- The **Hot Key** setting enables you to specify a key combination that users can type to launch the shortcut's target. It is recommended that you not use this setting, but instead allow users to define their own hot-key combinations.
- Any **Comments** you enter appear only in the Installation Designer and are not used in the installation.



BEST PRACTICE *Microsoft Best Practices guidelines require you not to create a shortcut in the root level of the Start menu, but instead to create the shortcut inside the Programs menu of the Start menu. If your product requires more than one shortcut, you should place the shortcuts inside a program folder (inside the Programs folder of the Start menu). Windows Installer installations automatically create a common or personal shortcut or program folder based on whether the installation is for all users of a machine or only for the current user (based on the Windows Installer property **ALLUSERS**).*

Creating Internet Shortcuts

The Windows Installer service does not automatically handle Internet shortcuts. An Internet shortcut is a text file with the .url extension, and you can create an Internet shortcut by installing a .url file with the appropriate contents.



TASK **For example, to create an Internet shortcut that launches the InstallShield Web site:**

1. Create a text file called `InstallShield.url` with the following contents:

```
[InternetShortcut]
URL=http://www.installshield.com
```

2. Place this file in a feature and install it as you would install any other type of file. When the end user launches the file, the specified Web site opens in the default browser.

Because a .url file is in the format of an .ini file, you can use the INI File Changes view of the Installation Designer (described later in this chapter) to create a .url file and modify its contents.

After an advertised shortcut is installed, its property sheet in the operating system appears as follows. Note that the target and icon cannot be changed for an advertised shortcut.

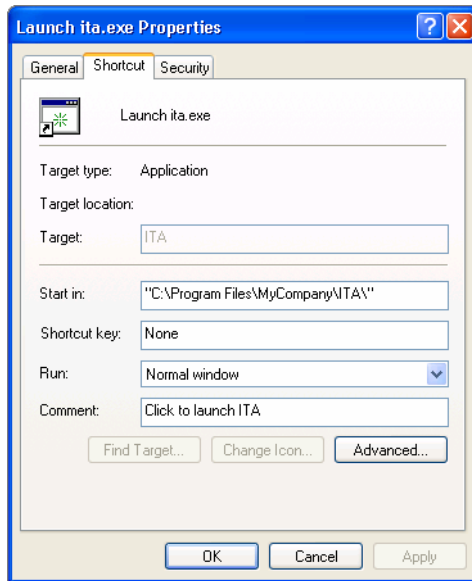


Figure 5-2: The Properties dialog box for the ITA shortcut.

Working with Registry Data

After installing files and shortcuts, the most common requirement for an installation program is to create registry keys and values on the target system. With Windows Installer, every registry key or value must be associated with an Express feature. The registry data is written to the target system when the feature is installed and removed from the target system when the feature is uninstalled.



TIP

To see all the registry data contained in your project—and not just the data associated with a single feature—select “All Application Data” from the Feature list at the top of the Registry view. As the name suggests, you can modify or delete existing registry data from the project. When creating new registry keys and values, however, you must explicitly select the feature to contain the new data.

Adding Registry Data

You can add registry data to a feature in the Registry view, under the “Configure the Target System” item in the View List.

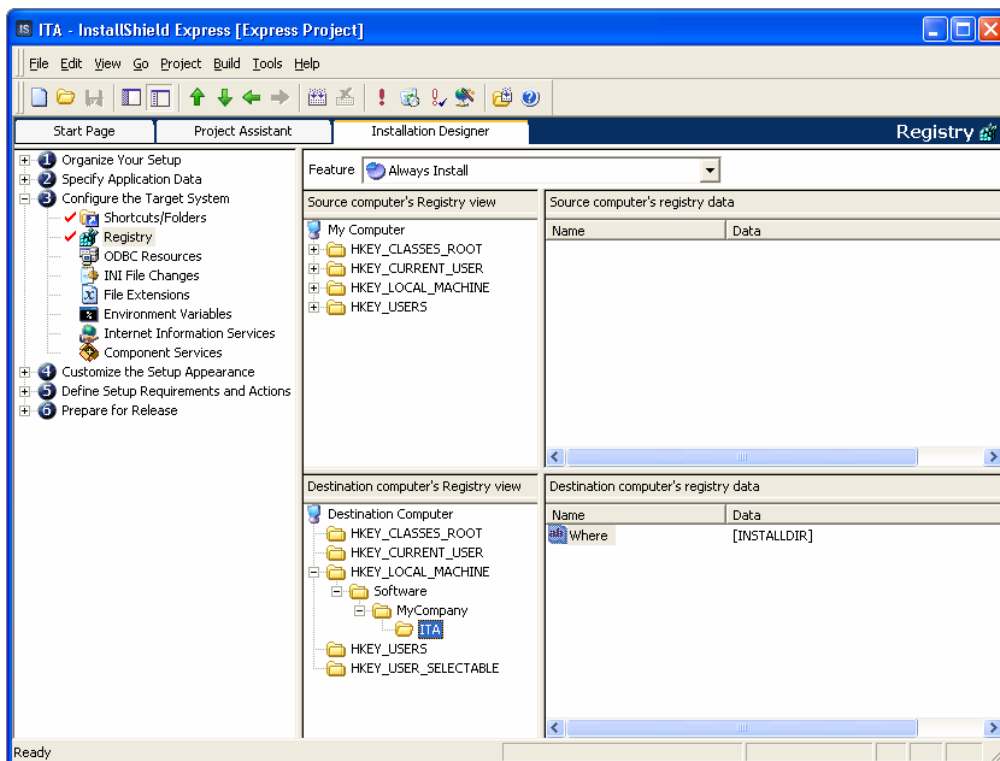


Figure 5-3: The Registry view in the Installation Designer.

Creating Registry Keys



TASK

To create a new registry key:

1. Select the desired feature from the Feature list at the top of the view.
2. Right-click the desired root key in the “Destination computer’s Registry view” pane (in the lower-left pane of the view) and select New Key.
3. Repeat for each subkey needed by the project.

After you have created a key, you can place values in it.

Creating Registry Values



TASK

To add a string value to an existing key:

1. Select the desired key in the “Destination computer’s Registry view” area.
2. In the “Destination computer’s Registry data” pane (lower-right section), right-click and select New String Value.
3. Enter the desired value name.
4. Double-click the value’s icon and enter the desired data in the “Value data” field.

Any registry data you create in the Installation Designer under the root key HKEY_USER_SELECTABLE will be created under HKEY_LOCAL_MACHINE (HKLM) for an all-users installation, and under HKEY_CURRENT_USER (HKCU) for a single-user installation. Note that HKLM and HKCU share only the Software subkey, so any data created under HKEY_USER_SELECTABLE should be placed under the Software key.

When the user runs your installation program, the registry data is written to the target system if the feature containing the registry data is selected for installation, and is removed from the target system when the feature is removed. When the end user uninstalls your product, Windows Installer removes only the registry data created by the installation program, and does not remove any keys or values created by your product after the installer ran. To specify that you want to remove all of the data contained under your registry key when the user uninstalls your product, you can right-click a registry key and select “Uninstall entire key.”



NOTE

To set the permissions for a registry key on Windows NT-based target systems, you can right-click a key in the Registry view and select Permissions, which displays a Permissions dialog box. The specifications you make in this dialog populate the LockPermissions table of your MSI database.



TUTORIAL For the ITA project, create the following registry entry:

1. Select Always Install from the feature list at the top of the view.
2. In the “Destination computer’s Registry view” pane, right-click HKEY_LOCAL_MACHINE and press **Insert**.
3. Type SOFTWARE\MyCompany\ITA for the registry key name. In Express, you can create several levels of registry key at once by separating the levels with backslashes. In this example, Express creates the SOFTWARE key, creates the MyCompany subkey under it, and then the ITA subkey under that.

4. Select the ITA folder icon, click in the “Destination computer’s registry data” pane, and press **INSERT**.
5. Type `Where` for the value name.
6. Double-click the `Where` value to display the Edit Data dialog. In the Value Data field, type `[INSTALLDIR]`.

If you run the installation and look in the registry of the target system, the `Where` registry value evaluates to the value of `INSTALLDIR`, as shown in the following figure.

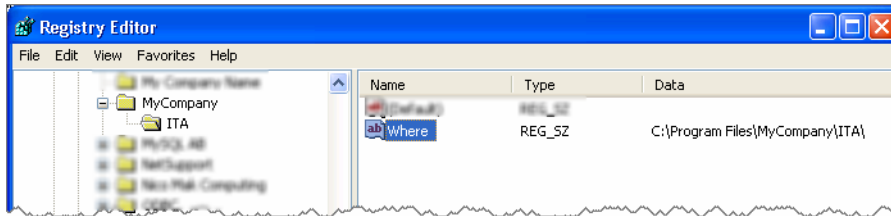


Figure 5-4: Value of `INSTALLDIR` in the registry.

Dragging and Dropping Registry Data

If the registry data you want to create on the target system exists on the development machine, you can drag the desired keys from the “Source computer’s Registry view” area into the “Destination computer’s Registry view” area. InstallShield Express supports different options when you right-drag (hold down the right mouse button and drag a key) a registry key from the source computer view to the destination computer view.

The options (“All keys and values,” “Key and its values only,” and “Only this key”) control whether subkeys and values are copied to the destination-computer pane. The default behavior—when you drag a registry key with the left mouse button—is to copy all the subkeys and values of the selected key into the destination-computer pane.

Importing a Registry File

Similarly, if you have registry data you want to create on the target system in a `.reg` file, you can right-click a key in the “Destination computer’s Registry view” pane and select **Import REG File**. If the data in the `REG` file is likely to change over time, you can use the feature’s “`REG File to Merge at Build`” setting instead, which re-imports your `REG` file data every time you build your installation project.

To create registry keys or values containing the values of Installer properties, you can use the syntax `[PROPERTYNAME]`, which expands to the value of `PROPERTYNAME` at run time. For example, to create a registry value that stores the

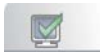
value of **INSTALLDIR**, set the value's data to `[INSTALLDIR]`, as you did earlier for the ITA installation project.

Application Paths

The application path registry key contains data that Windows uses as a private search path for the specified application's DLLs. If you install an application's DLLs into a directory not found in the PATH environment variable (and not into the application's directory), you should set the appropriate application path to include the DLL directory during installation. Application path information is stored in the registry under `HKLM\Software\Microsoft\Windows\CurrentVersion\App Paths\AppName.exe`.

Creating an Application Path

InstallShield Express makes it easy for you to create an application path from the Project Assistant.



TUTORIAL To create an application path for the ITA project:

1. Click the Project Assistant tab to open the Project Assistant.
2. Click the Application Registry icon to open the Application Registry page.
3. In the More Options section, click “Create an application path” to display the “Create Application Path” dialog.
4. Click the ellipsis (...) button to the right of the “Select the .exe file” field and browse to the `ITA.exe` file (located in `[ProgramFilesFolder]MyCompany\ITA`).
5. Click **Open** to add the file and the directory.

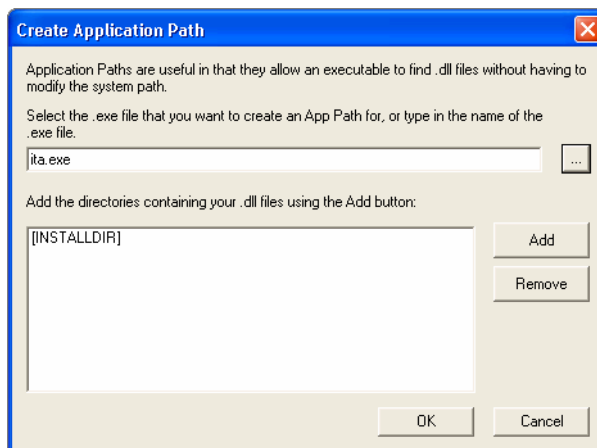


Figure 5-5: Creating an application path using the Create Application Path dialog.

If the DLLs in your installation will be installed in a different directory, you can add those directories by clicking the **Add** button and browsing to the directory.

Installing ODBC Resources

InstallShield Express provides an ODBC Resources view that displays all the ODBC drivers, translators, and DSNs installed on the development system. To duplicate the desired ODBC information on the target system, select the drivers, translators, and DSNs and specify their properties (such as whether a DSN is a user data source or a system data source), selecting the features to which you want to attach the data.

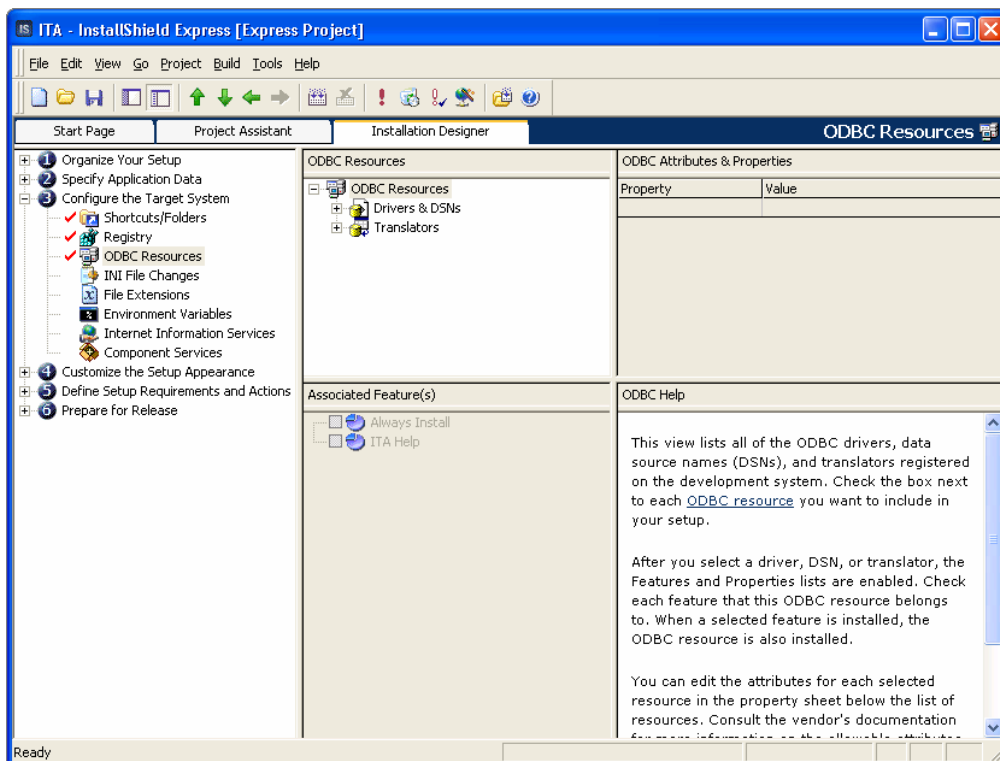


Figure 5-6: The ODBC Resources view in the Installation Designer.

Making Changes to INI Files

In the INI File Changes view, you can specify data to be written to an .ini file during installation. For this example, suppose you want to create a file called `sample.ini` in `INSTALLDIR`, containing the following data.

```
[BasicSettings]
ThisComputer=TargetComputerName
```

The italicized data will be replaced with the proper values during installation.



TASK To create an INI file (or specify an existing INI file):

1. Right-click the INI Files icon, select Add INI File, and enter the desired file name.
2. In the file's property sheet, select the feature with which to associate the INI file, and select the directory on the target system in which the INI file will be created or located.

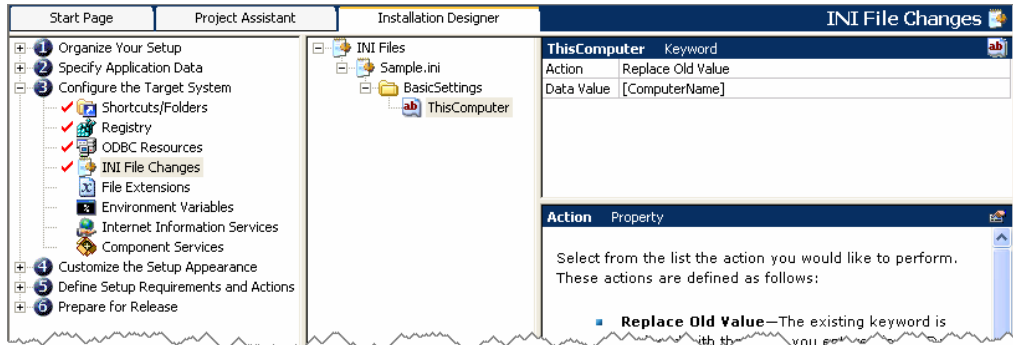


Figure 5-7: Adding an .ini file in the INI Files view.

3. To create a section in the INI file, right-click the file name icon and select Add Section, typing the section name without the surrounding square brackets.
4. To add keys and values to a section, right-click the section icon and select Add Keyword.

As with registry data, you can expand the values of installer properties in the .ini file using the syntax `[PropertyName]`.



TUTORIAL For the ITA project, make the following changes in the INI Files view:

1. Right-click the INI Files icon and select Add INI File, or press **Insert**.
2. Name the file `Sample.ini`. You can leave the default settings; the Feature setting should be Always Install and the Target setting should be `[INSTALLDIR]`.
3. To add a section, right-click the `Sample.ini` icon and select Add Section, or press **Insert**. Name the section `BasicSettings`.
4. To add a keyword, right-click the `BasicSettings` section and select Add Keyword. Name the keyword `ThisComputer`.
5. For the Action setting, select “Replace Old Value.”
6. For the Data Value setting, type `[ComputerName]`. (**ComputerName** is a private Windows Installer property that holds the computer name of the target system.)

(Information that you enter in the INI File Changes view is written to the IniFile table of the built MSI database.)



TIP

You can import an existing INI file into the INI File Changes view by right-clicking the INI Files icon and selecting Import INI File.

Creating File Extensions

File name–extension associations, or file associations, are registry settings that tell Windows what application to use to open files of a certain type. For example, Windows typically opens text files (files with the .txt extension) with Windows Notepad, and opens bitmap files (files with the .bmp extension) with Microsoft Paint.

On Windows NT 4 and Windows 9x, file association information is stored under the root key HKEY_CLASSES_ROOT. On Windows 2000 and later, file associations are stored in both HKLM\Software\Classes and HKCU\Software\Classes; you see a merged view of the data under HKEY_CLASSES_ROOT.



BEST PRACTICE *Windows Installer Best Practices guidelines recommend that you create a file association for every non-hidden type of file created or used by your product.*

From within Windows Explorer, you can view and modify registered file types in the File Types tab of the Folder Options property sheet. (To view this property sheet, select Folder Options from the Tools menu in Windows Explorer.)

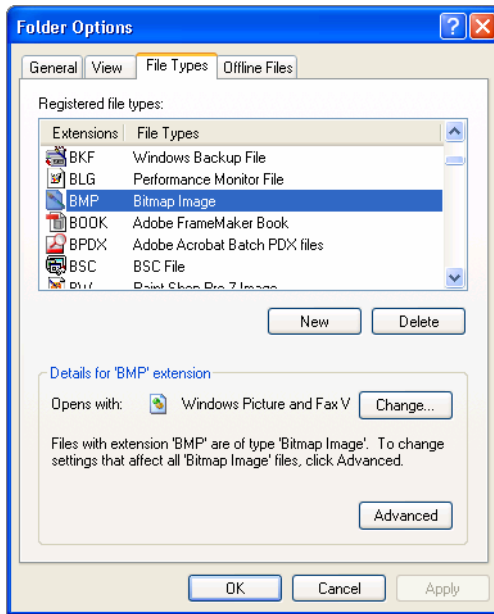


Figure 5-8: View registered file types in the Folder Options property sheet.

Similarly, you can view the application associated with a given file by right-clicking the file icon and selecting Properties.

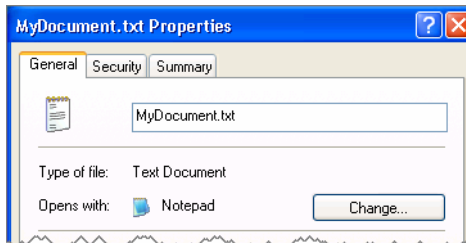


Figure 5-9: Viewing the application association for a file.

In an Express installation project, you can use the File Extensions view to create a file association.

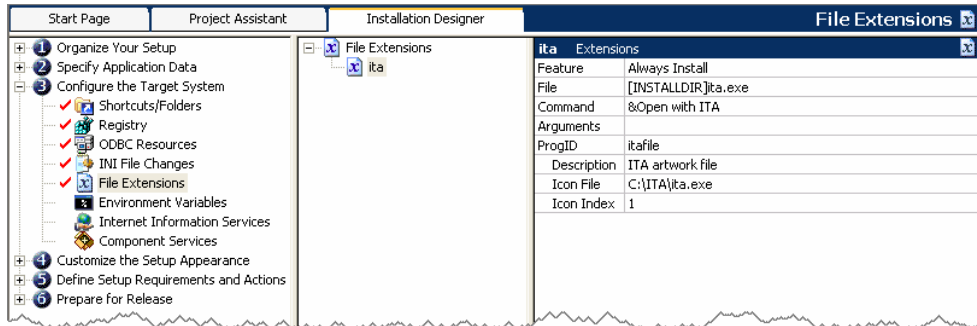


Figure 5-10: The File Extensions view in the Installation Designer.




TUTORIAL To create the association for the ITA file type:

1. Open the File Extensions view in the Installation Designer.
2. Right-click the File Extensions icon and select New Extension, or press **Insert**.
3. Rename the file name extension without a leading period (*ita*, not *.ita*). Express automatically creates an Open command, used by the operating system to open files with the given extension.
4. Make sure that the Feature setting for the file extension is set to Always Install.
5. For the File setting, select **[INSTALLDIR]ita.exe** from the drop-down menu.
6. In the Command field, type **&Open with ITA**. This sets the text that appears when a user right-clicks on a file with the *.ita* file extension (see the following figure).
7. In the ProgID field, type *itafile*. Information about ProgIDs is provided in this chapter.
8. In the Description field, type *ITA artwork file*. This is the description that you enter here is used in the File Properties dialog box, which is displayed when the end user right-clicks on a file with the specified file extension and selects Properties.
9. In the Icon File field, browse to the *ita.exe* file. In the Change Icon dialog that is displayed, select the icon that you want to display for this file type. This automatically populates the Icon Index setting.

The ITA application creates, saves, and opens files with the *.ita* extension.



TUTORIAL To create a document with the .ita extension:

1. Run the ITA installation by clicking the Run button () on the toolbar.
2. Launch ITA by selecting the application from the Programs menu.
3. Left-click and right-click in the main window to create your artwork.
4. To save an ITA file with the name `Testing.ita`, select “Save as” from the File menu, name the file `Testing`, and save the file to your desktop.

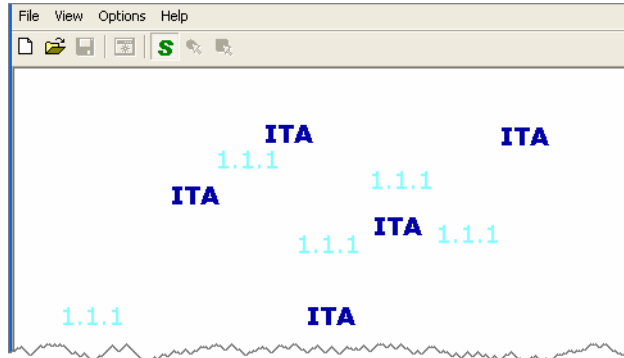


Figure 5-11: Creating artwork with the ITA application.

5. When you right-click on the `Testing.ita` file, the Open command presents ITA as the application with which it will open the file, as shown in the following figure.

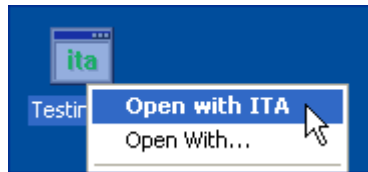


Figure 5-12: Choosing to open `Testing.ita` with the ITA application.

Using Arguments

In the Arguments field, you can provide any command-line arguments that you want to pass to your application when a file of this type is opened. The argument `%1` acts as a placeholder for the name of the file that the user double-clicks. For example, `-p %1` would resolve to `-p C:\MyFile.ext`.



NOTE

In some cases, it is necessary to enclose the `%1` argument with quotation marks—as in “%1”—to correctly handle file names that contain spaces.

Specifying a ProgID

For each extension you define, you must create or specify a ProgID (sometimes called a file type's application identifier or tag name). A file type's ProgID is an arbitrary string, but it should be unique on the target system. One ProgID naming convention is to append the word *file* to the extension without a dot—the .ext extension might use the ProgID *extfile*. Another convention is to name a file-type ProgID after the application used to open the file type, as in *ITA.Document*. To create a ProgID, enter the ProgID name in the extension's ProgID field (as in the value *itafile*).

For each ProgID, specify the file-type description and its icon file and index. (The path you specify to the icon file is the path on your development system, and not a directory on the target system.)

After running the installation, a user can double-click a file created with the .ita extension, and Windows opens it with InstallShield Training Application (ITA). When a user right-clicks `Testing.ita` and selects Properties, information similar to the following is displayed, showing the file-type icon and description.

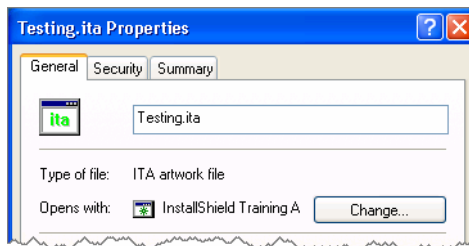


Figure 5-13: Viewing the file type and the association.



TIP

For a faster way to create a file extension, you can use the Application Shortcuts page of the Project Assistant, as described in Chapter 2.

Working with Environment Variables

Environment variables are strings that contain information about the user or about the environment, including drives, paths, or file names. These variables are usually set during login, and can be either user-specific or system-specific. Any user can change user environment variables, however, only an administrator can add, remove, or modify a system environment variable.

To create or modify environment variables on the target system, you can use the Environment Variables view of the Installation Designer, located under the “Configure the Target System” section in the View List. As with other types of data, environment variable changes must be associated with a feature, and (by default) the

variable data are installed when the feature is installed, and removed when the feature is removed.

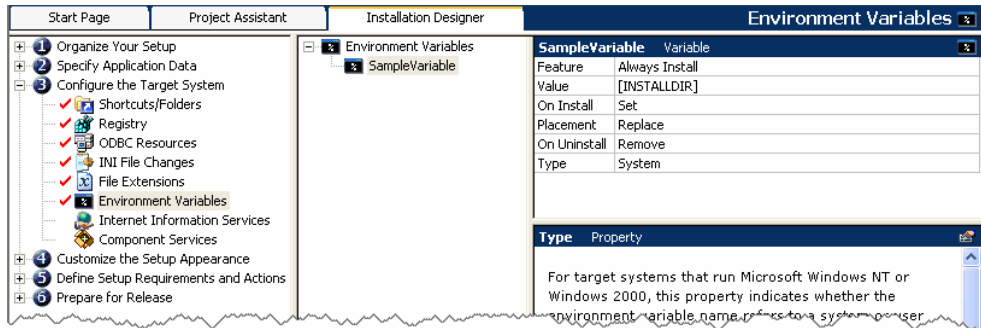


Figure 5-14: The Environment Variables view in the Installation Designer.

NOTE

When possible, it is recommended that an installation program not be used to modify the user's PATH environment variable, but instead to create a per-application path for the product's executables. Per-application paths were described in the previous section.

TUTORIAL To add environment variable data to the target system when ITA is installed:

1. In the Environment Variables view, right-click the Environment Variables icon and select Add Environment Variable, or press **Insert**. Name the environment variable `SampleVariable`.
2. In the Feature field, select the feature with which you want to associate the variable—in this case, the Always Install feature.
3. In the Value field, enter the value to write to the variable. As with registry and INI-file data, you can expand the value of a Windows Installer property using the format `[PropertyName]`. Type `[INSTALLDIR]`.
4. In the On Install field, select Set. This indicates that the environment variable will be created (if it does not already exist on the target system) and set to the value specified in the Value field.
5. In the Placement field, select Replace. (If you intend to modify the PATH variable or any other existing environment variable, you should select to append or prepend, and not replace, the existing variable data.)
6. In the On Uninstall field, select Remove. This indicates that you want the environment variable removed when the application is removed.
7. In the Type field, select System to create a system environment variable to indicate that you want the environment variable to be available to any user logged on to the target system. If User is selected, Express creates a user environment variable, which is available only to the user who ran the installation program. (The

distinction between system and user environment variables is recognized only on Windows NT–based systems; this setting is ignored on systems running Windows 9x.)

On a Windows 9x system, environment variables are stored in `Autoexec.bat`; on Windows NT–based systems, system environment variables are stored in the registry key `HKLM\System\CurrentControlSet\Control\Session Manager\Environment`, and user environment variables are stored in `HKCU\Environment`.



TUTORIAL To see the `SampleVariable` environment variable:

1. Run the installation project by pressing the Run button on the toolbar.
2. From the Control Panel, open the System Properties panel.
3. Click the Advanced tab.
4. Click **Environment Variables**.

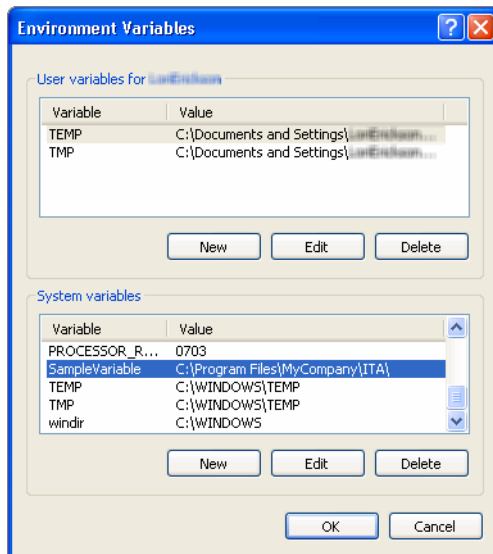


Figure 5-15: Viewing the `SampleVariable` environment variable on the target system.

Installing Special Types of Files

For many files, your installation program’s only requirement is to copy the files from the source media to the target system. For some types of files, however, the installer also needs to register the files with the target system. In this section, you will see how to install fonts and .NET applications on a target system.

Installing Fonts

In order to install a font file (.ttf, .ttc, or .fon file) correctly, an installer must write font information into the registry key HKLM\Software\Microsoft\Windows NT\CurrentVersion\Fonts (on Windows NT-based systems) or HKLM\Software\Microsoft\Windows\CurrentVersion\Fonts (on Windows 9x), in addition to transferring the font files.



TASK To install a font file using an Express project:

1. Open the Files View.
2. From the menu at the top of the view, select the feature with which you want to associate the font file.
3. In the “Destination computer’s folders” pane, right-click the Destination Computer icon, select Show Predefined Folder, and select [FontsFolder].
4. Select the [FontsFolder] icon and drag the .fon or .ttf file from the “Source computer’s files” pane to the “Destination computer’s files” pane.



LOGO

The “Designed for Microsoft Windows” Logo guidelines allow you to leave any installed fonts on a target machine after the user removes your application. To leave a font on a machine after uninstallation, select the Permanent setting of the font file (in the Advanced tab of the File Properties dialog box).

Installing .NET Applications

Windows Installer version 2.0 introduces support for installing .NET applications, with new MSI database tables, properties, and actions. .NET modules (EXEs and DLLs) are typically represented by assemblies. An assembly is a module with extra information describing the module’s dependencies and properties; the part of an assembly that describes the module’s dependencies is called a manifest.

InstallShield Express Edition provides build settings that allow you to add .NET support to your installation project.

As described in Chapter 8 of this manual, “Building Releases,” if your installation project uses .NET features, you should select to set the .NET Framework properties in the Build Your Release view. Note that .NET features require version 2.0 of the MSI engine. The .NET Framework is not supported on Windows 95.

When you build a release for a project that uses .NET properties but does not include the .NET Framework redistributable, build warning –6245 is displayed: “One or more of the project’s features contain .NET properties that require the .NET Framework. It is recommended that the release include the .NET Framework.”

.NET File Properties

The following file properties are related to installing .NET applications and are located in the COM & .NET Settings tab of the File Properties dialog.

- **Registration Type:** Specify whether or not this file supports self-registration or if it extracts COM information. Possible values are None, Extract COM Information, and Self-registration.
- **Scan at Build:** Specifies what data, if any, Express should scan in your .NET application during a build; possible values are Dependencies and Properties, Properties Only, and None.



NOTE

You can specify the default .NET Scan at Build file setting by pulling down the Tools menu, selecting Options, selecting the .NET tab, and selecting the desired default from the “Default .NET Scan at Build File Setting” list.

- **Application File:** Specifies the file that the **Scan at Build** setting should scan.
- **Installer Class:** Select this option to ensure that during installation the assembly’s Install, Commit, Rollback, and Uninstall methods will be called at the appropriate time.
- **COM Interop:** Select this option if your .NET application supports COM interoperation (the ability to call .NET objects from COM). For example, if you have a Visual Basic.NET class library that defines the ComClass attribute and contains at least one public (COM-callable) function, Express extracts the COM Interop information at build time and adds it to the Class and ProgID tables of your MSI database.

Instead of scanning an assembly’s dependencies during each build, you can use the Static Scanning Wizard (located in the Dependencies view). If the Static Scanning Wizard detects any assemblies required by an assembly that exists in the project, it offers to add the detected assemblies to your project. Similarly, you can create InstallShield Express Edition projects inside Microsoft Visual Studio .NET.

Installing an Assembly to the Global Assembly Cache

By default, any dependent assemblies detected with the Dependencies and Properties setting (specified from the Scan at Build setting) will be installed to the same directory as the assembly that requires them. If an assembly will be required by multiple applications, you might want to install the assembly to the Global Assembly Cache (GAC).

The GAC is a special folder (typically located at `C:\WINNT\Assembly` or `C:\Windows\Assembly`) that stores shared assemblies. What makes the GAC special is that multiple versions of an assembly with the same file name can be installed into

it, so different applications can use a known good version of any assembly that it requires.



NOTE

To install an assembly to the GAC, the assembly must have a strong name. You give an assembly a strong name by generating a public/private key pair; using, for example, the .NET Framework SDK tool `SN.exe`, and then applying the key pair to the assembly.

To generate the key pair, run a command like the following:

```
SN -k SampleAppKeyPair.snk
```

This command creates a binary file containing the public and private key numbers. You can then add the assembly attribute `AssemblyKeyFile` to your application source code, as in the following C# code:

```
[assembly: AssemblyKeyFile("SampleAppKeyPair.snk")]
```

In Visual Basic.NET code, the attribute appears as follows:

```
<Assembly: AssemblyKeyFile("SampleAppKeyPair.snk")>
```

After you build the assembly, it will have the strong name required for installing it into the GAC.

Assuming your assembly has a strong name, you install it into the GAC by using [GlobalAssemblyCache] as your component's Destination property.

After building and running your installation project, your assembly will appear in the GAC.



NOTE

*Windows Installer version 2.0 introduced the property **MsiNetAssemblySupport**, with which you can determine if a target system supports .NET assemblies.*

Internet Information Services

In addition, the “Configure the Target System” section in Express contains an Internet Information Services view, in which you can define and configure virtual directories to create on a target system having IIS 4, IIS 5, or IIS 6 present.

During installation of a package that includes virtual directories, InstallShield Express checks for the existence of IIS on the target system. If IIS is not installed, the installation displays a dialog informing the end user that they do not have IIS installed and asking whether they want to install and continue. If the end user does not install IIS, the installation continues, but the virtual directory is not created.

Component Services

Express also provides a Component Services view, which enables you to create or modify COM+ applications on a target system, using a development environment similar to the Component Services administrative tool in Windows 2000 and later. In the Component Services view, you can manage COM+ server applications and components for your installation. The component services (currently only COM+ applications) that are installed on the development machine are listed under the COM+ Applications icon.