Seriously Flexible Installation Software

User's Guide

IndigoRose
SOFTWARE DESIGN CORP.

http://www.indigorose.com
info@indigorose.com

Proprietary Notice
The software described in this document is a proprietary product of Indigo Rose Software Design Corporation and is furnished to the user under a license for use as specified in the license agreement. The software may be used or copied only in accordance with the terms of the agreement.

Information in this document is subject to change without notice and does not represent a commitment on the part of Indigo Rose Software Design Corporation. No part of this document may be reproduced, transmitted, transcribed, stored in any retrieval system, or translated into any language without the express written permission of Indigo Rose Software Design Corporation.

Trademarks
Setup Factory, TrueUpdate, Visual Patch and the Indigo Rose logo are trademarks of Indigo Rose Software Design Corporation. All other trademarks and registered trademarks mentioned in this document are the property of their respective owners.

Copyright

Indigo Rose Corporation
11/2001

Table of Contents

Chapter 1

Introduction

Welcome to the Setup Factory User's Guide. This User's Guide is designed to explain important concepts and help familiarize you with the features available in Setup Factory.

What is Setup Factory?

Setup Factory is a sophisticated tool that gives you complete control over the installation process. Full-featured, fast, and easy to use, Setup Factory's intuitive design maximizes your ability to deploy software products, data files, graphic images, or anything else that you want to distribute.

Solid and Effective

Setup Factory 6.0 creates single-file, compact, *bulletproof* professional software installations. Its visual design environment makes it a snap to assemble your files, customize the wizard interface, add runtime support modules, and package your software for deployment via diskette, CD-ROM, LAN, email or web.

A Proud Heritage

Since its first release, Setup Factory has defined innovation in Windows based setup and installation tools. Over the years, the product line has been the recipient of numerous awards, accolades and glowing reviews.

Trusted By Professionals

Thousands of software developers trust Setup Factory to distribute their software to millions of customers and clients around the world.

While others have tried to imitate it, only Setup Factory 6.0 offers such a perfect combination of ease, flexibility and control.

Installers and Setup Files

Installers are special self-contained executable programs designed to transport whole software products to your users and deploy them on their systems. Also known as "setup files," "setups" and "setup executables," professional installers like Setup Factory are able to fully configure the user's system to perfectly accept their software cargo.

In order to adapt to field conditions, installers must be able to accurately gather information on the software's operating environment. This can involve anything from analyzing the amount of free space available on a hard drive, to ensuring that the user's system is properly configured to accept the software being installed.

In order to support the widest possible markets, installers need to interact with users in a streamlined, sensible fashion. The best installers do this by presenting a familiar, user-friendly, wizard style interface. In the case of Setup Factory, not only is the user interface friendly and familiar, but it's fully customizable to meet any need as well.

What's New in 6.0?

Actions

Only Setup Factory could integrate powerful scripting control into its visual design environment in such a seamless manner. While most users will never need to go beyond the visual side of Setup Factory, developers who require absolute control will appreciate the extremely flexible "Action Scripts" built into Setup Factory 6.0. Actions give you the flexibility of advanced programming, without sacrificing the ease of use that Setup Factory is famous for.

Control Structures

Build advanced looping and testing structures with actions like IF, WHILE and GOTO Label.

Variables, Expressions and Operators

Use an unlimited number of variables to store data. Throw in full expression parsing and mathematical evaluation, and you've got enough power to make your installer handle anything you throw at it!

String Functions

Parse, evaluate, manipulate and search through text files, text variables, and other strings.

Silent Installations

The /S command line option lets you perform installations without user interaction.

Download Files and Upload Data

Make it easy for your users to stay up-to-date. Built-in HTTP download and web data submission make it easy to distribute files "after the fact."

Enhanced Package Support

Package support has been completely redesigned, and sports a new, integrated design interface. By assigning files to packages, you can give your users greater control over what files are installed on their system. You can even present your users with multiple levels of control, including the standard Complete, Typical, Minimum, and Custom installation options. And there's more good news for power users: now files can be assigned to multiple packages!

File Dependency Scanning

Quickly scan Portable Executable files to analyze their runtime dependencies. With a negative import option, you can control which dependency files are added to your project.

ZIP File Extraction

Use the Unzip Files action to extract files from a Zip archive at run time.

Full Control over Windows Services

Setup Factory 6.0 adds full support for Windows services, including starting, stopping, pausing, querying, creating, continuing, and deleting them.

Full Control over Programs and Processes

With full shell execution support, starting or terminating programs and processes is a piece of cake.

Enhanced 3rd Party Runtime Support

Instantly add runtime support for many popular third-party technologies.

Enhanced Visual Basic Project Support

Core runtime support and advanced Visual Basic project file scanning make creating installers for Visual Basic applications easier than ever.

Updated Run-time Interface

The setup executable's user interface has been updated to the professional and modern "Windows Installer" style.

Design-Time Constants

Similar to variables, but instead of being converted to values at run time, design-time constants get converted at build time. Like #define statements in C++, you can use design-time constants to simplify your project management.

Enhanced Reporting and Logging Features

Keeping track of essential project details has never been easier. With customizable HTML-based project reports and text-based install-time log files, you'll have an accurate record of everything you need.

Key Features

Award-Winning Visual Design Environment

Organize your project quickly and efficiently using Setup Factory's innovative, point-and-click development environment.

Quick and Easy Project Wizard

The Project Wizard is the fastest and easiest way to get your installation off to the right start. With the Project Wizard you'll have an installation ready to go in minutes.

Dynamic File Lists

Set up your project the way you see fit. When you're ready to build the setup executable, your project is rescanned to include the most recent versions of your files.

Flexible OS Filtering

Target multiple operating systems with one installer. Sophisticated condition and expression parsing lets you filter files and actions to specific operating systems.

Flexible Deployment Options

Distribute on any media size or format, including web, email, ftp, CD-ROM, DVD, network drive or even 1.44MB floppy disks.

Registry and System Modifications

A full suite of actions are supplied for reading and writing information to and from the Windows Registry, INI files and text configuration files.

OCX/DLL and Font Registration

Register TrueType fonts and OCX/DLL files right from your installer.

Extensive File Handling Options

Complete version checking, file searching, copying, renaming, deleting, attributes setting, and more.

International Language Support

Setup Factory has built-in handling for every language supported by Windows.

Comprehensive Operating System Support

You don't want to mess around with reliability. Your install will run on all 32 bit Windows platforms. This includes everything from Windows 95, through 98, ME, NT4, 2000 and even Windows XP.

Getting the Most from this User's Guide

The Setup Factory 6.0 User's Guide is divided into 19 chapters and 4 appendices. We've tried to organize these chapters in a way that will benefit new and experienced users alike. The first five chapters cover important concepts, introduce you to Setup Factory's design environment, and teach you the first steps in designing an installer. Chapters 6 through 13 serve as a reference to the core parts of Setup Factory's design environment. Chapters 14 through 19 cover advanced topics and offer insights and advice on building, testing and distributing your installer.

New Users

If you're new to installers, or new to Setup Factory, reading this User's Guide from front to back will teach you everything you need to know to become proficient at building installers for your software. If you're eager to get started, then Key Concepts (page 25), Getting Started (page 45), and Quickstart Tutorial (page 71) are the three "must-read" chapters you'll want to read in order to get going as quickly as possible.

Experienced Users

If you're an experienced Setup Factory user, you probably want to jump right in and see the new features in action. In that case, feel free to use this guide as a reference. When you encounter a feature that you'd like to know more about, use the Index and the Table of Contents to quickly locate information about that feature in this User's Guide.

Of course, even advanced users will gain valuable insights by reading this User's Guide from cover to cover. It's a good idea to skim through the Key Concepts chapter (page 25) and look for concepts that you aren't already familiar with. Design-time constants (page 37) and primer files (page 43) will certainly be new to you. It's also wise to read the chapter on Expressions (page 221) that covers the intricacies of the conditional expressions that you can use in Setup Factory 6.0.

Advanced users might also want to browse the list of actions in Appendix A (page 263), and the list of built-in variables in Appendix B (page 269).

Setup Factory is an advanced application, and there's a lot of information in this User's Guide. We hope you'll find this User's Guide as easy to use as Setup Factory itself.

Document Conventions

Throughout this User's Guide, we've presented different kinds of information in the following ways:

Start Menu

Items in the Start menu are presented in bold, italicized text and separated by -> symbols, like so:

Start -> Programs -> Indigo Rose -> Setup Factory -> Important Notes

To access this item, you would click on the ***Start*** button, click on ***Programs*** in the Start menu to open the Programs menu, click on ***Indigo Rose*** in the Programs menu to open the Indigo Rose menu, click on ***Setup Factory*** in the Indigo Rose menu to open the Setup Factory menu, and click on the ***Important Notes*** menu item.

Setup Factory Program Menus

Items in the Setup Factory program menus are presented in bold text and separated by | symbols, like so:

Project | Build

To access this item, you would first click on the **Project** menu at the top of the Setup Factory program screen to open the Project menu, and then click on the **Build** item in that menu.

Paths and filenames

Directory paths and filenames are presented in a monospace font, like so:

```
C:\Program Files\Setup Factory
```

This path refers to the Setup Factory folder within the Program Files folder on your C: drive.

Tips, Notes, Warnings and Cross-References

We've used special formatting to set some information apart from the rest of the text:

TIP

Tips provide helpful information, such as alternative ways of accessing a feature or shortcuts for advanced users.

NOTE

Notes provide extra information to clarify points discussed in the text.

IMPORTANT

! Warnings alert you to important information.

SEE ALSO

? This is how we highlight links to information online, in the Command Reference or on other pages in this User's Guide.

Dialogs and Screens

The names of Setup Factory dialogs and the names of screens that you can add to your installer are both presented in italics:

The *General Design* dialog
The *Select Packages* screen

Buttons

The names of Setup Factory buttons are presented in bold text, like so:

The **Properties** button

When applicable, button names are followed by a thumbnail of the button image between parentheses:

The **Add** button ()

Fields

The names of fields and other GUI elements like check boxes and drop-down lists are presented in bold text, like so:

Select the **Evaluate value as expression** check box

Other Resources

If you ever have any questions about the software that aren't answered by this User's Guide, there are several other resources available to you:

Command Reference and Online Help

The Command Reference is included in HTML Help format. It is accessible from both ***Start -> Programs -> Indigo Rose -> Setup Factory -> Help Contents*** and from the **Help** menu within the product. This is a comprehensive reference for all the Setup Factory screens, options and properties.

TIP

Most screens within the development environment contain a **Help** button, which will automatically open the appropriate topic in the Command Reference.

Web Site

The Setup Factory web site is a great place to learn about the product. It is located at http://www.indigorose.com/setup/.

The web site is where you will find:

- Knowledge Base articles
- Tutorials
- Answers to frequently asked questions
- Information about the latest version, bug fixes, etc.

Forums

We maintain a number of popular web-based discussion and support forums at http://www.indigorose.com. These forums are an ideal place to meet with other users of Indigo Rose products. Members frequently discuss the usage of products, share tips and tricks, exchange ideas and much more.

Technical Support

For information on tech support please see page 280.

Chapter 2

Key Concepts

This chapter explains important concepts and terminology that you must understand in order to become proficient with Setup Factory.

TIP

If you're an experienced Setup Factory 5.0 user, you might want to jump directly to the explanation of Actions on page 34.

Design Time

Design time refers to the process of designing your setup using the Setup Factory design environment. Everything you do with Setup Factory, from the moment you start the program, to the point where you are ready to actually generate the executable setup file, is considered work done "at design time."

Design time is where you, the developer, make the choices that will determine what your setup will do, how it will do it, and what it will look like when it is done.

Build Time

Build time refers to the final step of actually generating the setup executable file. This is where Setup Factory takes all the work you did at design time and converts it into the final product that your users will run.

Run Time

Run time refers to when the actual setup executable is run. This is where the interface you customized during design time is presented to the user, and the actual work of installing the software is performed.

Releases

A release is defined as a set of files that are distributed as a whole unit. In other words, a release consists of all the files that make up one version of your software.

In Setup Factory, a release consists of all the files that you need to install on the user's system in order for them to be able to use your software.

Files, Folders and Paths

In order to be able to design your installer, it's important that you have a solid understanding of files, folders and paths. If you're new to Windows and hierarchical file systems, you should pay extra attention to this section.

Files

Information on computers is stored in files. A file is a collection of computer-readable data that is treated as a single entity by the computer's operating system. In other words, a file is a bunch of data that a computer can store as one unit.

Files vary in size and can contain all sorts of data, from text, images and sounds, to database records and the machine language code that programs are made of.

Each file is identified by its filename and path. The filename is the name that was given to the file when it was created or saved. The path is a string of text that describes where the file is stored.

Extensions

A file extension consists of a period (.) followed by one or more characters at the end of a filename. Windows uses the file extension to determine what kind of information is contained in a file. For example, in the filename `myfile.txt`, `.txt` is the file extension that identifies `myfile.txt` as a text file.

TIP

When reading filenames out loud, the period in the extension is usually pronounced "dot." So `myfile.txt` would be pronounced "myfile dot text" or "myfile dot tee ex tee."

Folders

Windows uses a *hierarchical* file system. This means that storage space is organized into folders and sub-folders, forming what is often referred to as a "directory tree." The base of the directory structure is known as the *root folder*, or just the "root" for short. Within this root folder there can be files and sub-folders, and within those sub-folders there can be other files and sub-folders as well.

Each folder acts like a container for all the files and folders that are "in" it. All the files and sub-folders in a folder can be copied, moved or deleted at once by copying, moving or deleting the folder that contains them.

Much like hanging folders in a filing cabinet, folders make it easier to organize and locate your files. If the files on a computer were all kept in one place, it would be very difficult to find a particular file. Folders allow related files to be grouped together so it's easier to find them.

NOTE

Folders are also often called directories. In Windows, the terms "folder" and "directory" both refer to the same thing.

Drives

A drive is a form of fixed, networked or removable media used as a storage device. A "fixed" drive is one that isn't removable—which is to say that the storage media is built right into the drive, and isn't made to be removable like floppy disks and CD-ROMs are. A "networked" drive is a one that isn't connected locally, but is instead accessed remotely via a network. A "removable" drive stores information on removable media such as magnetic or optical disks, cartridges and tapes.

The term "drive" can also refer to the drive letter, which is the letter assigned to the drive during the computer boot process. A single letter of the alphabet is assigned to each drive as it is detected at startup. The letters "A" and "B" are reserved for floppy drives, and the rest of the letters (from C to Z) are assigned in order to the various hard drives, CD-ROM drives, and any other drives attached to the system.

For example, if your system has a single hard drive and a CD-ROM drive, your C: drive is your hard drive, and your D: drive is your CD-ROM. If your hard drive is split into two logical drives or "partitions," then C: and D: would refer to those partitions, and your CD-ROM drive would be E:.

NOTE

Drives are where files and folders are stored.

Root Folder

The root folder is the "base" or "main" folder on any drive. The root folder is where all of the other folders on a drive are located. When you double-click on the C: drive in `My Computer`, you're opening the root folder of the C: drive. All the folders on the C: drive are located in the "root" of C:.

The root folder is always named "\" in Windows. For example, the path to the root of the D: drive is:

```
D:\
```

Paths

A path is a string of text that describes where a file or folder is stored in a hierarchical directory structure. There are three kinds of paths you can use in Setup Factory: full paths, relative paths and UNC paths.

Full Paths

Full or "absolute" paths provide "complete" directions to locate a file, starting from the root folder of a given drive. A full path begins with the drive letter and includes

the name of each folder that would need to be opened, in turn, in order to access the desired folder or file. The folder names are separated by backslash characters (\). Full paths have the following general format:

<drive letter>:\<folder name>\<folder name>\<filename>

For example:

- The path to the root folder on the C: drive is:
  `C:\`
- The path where Setup Factory 6.0 is installed on your system is probably:
  `C:\Program Files\Setup Factory 6.0`
- The path to `Notepad.exe` on your system is probably something like:
  `C:\Windows\Notepad.exe`

Relative Paths

Relative paths provide "partial" directions to locate a file starting from another folder (often the current working directory). They look just like full paths, but they're missing the drive name and possibly some folder names too. The simplest relative paths consist of a single folder or filename. Relative paths have the following general format:

<folder name>\<folder name>\<filename>

For example:

- The relative path to a Data folder in the current working directory would be:
  `Data`
- The relative path to the Setup Factory 6.0 application from the `Program Files` folder would be:
  `Setup Factory 6.0\SUF60Design.exe`
- The relative path to a readme file in a sub-folder named `Docs` would be:
  `Docs\readme.txt`

UNC paths

The Universal Naming Convention (UNC) is a standard method of describing the location of files and other resources shared on a network. In Windows, UNC paths begin with two backslashes (\\), followed by the *server name*, which is the name assigned to the computer where the shared resources are located. The server name is followed by another backslash (\) and the *share name*, which is simply the name that was given to the volume or storage device when it was shared. This is then followed by the path to the desired file or folder on that shared volume.

UNC paths have the following general format:

\\<server name>\<share name>\<folder name>\<filename>

For example:

- The UNC path to the Setup Factory 6.0 application on a C: drive which is shared as "D2" on a computer named "R2" would be:
  `\\R2\D2\Program Files\Setup Factory 6.0\SUF60Design.exe`

- The UNC path to a file named `foo.txt` in the `temp` folder of a drive which is shared as "MAIN" on a computer named "DOROTHY" would be:
  `\\DOROTHY\MAIN\temp\foo.txt`

Shortcuts

A *shortcut* is a very small file in the Windows operating system that points or "links" to a file or web site.

Each shortcut file contains information about where the file or web site that it "points" to is located. When a user double-clicks on a shortcut file, the file or web site that it points to is opened instead.

Shortcut files have a `.lnk`, `.url` or `.pif` extension that is hidden by the Windows operating system.

Items in the Favorites menu and the Start menu are all shortcuts. These shortcuts are usually organized into folders, known as *shortcut folders*. A shortcut folder is simply a folder in the Start menu that contains shortcut files.

CRC Values

CRC values are calculated using an algorithm known as the Cyclic Redundancy Check, or "CRC" for short. Basically, this involves generating a 32-bit number (or "CRC value") based on the contents of a file. If the contents of a file change, its CRC value changes as well. This allows the CRC number to be used as a "checksum" in order to identify whether or not the file has changed. It also allows you to distinguish between different versions of a file by comparing its CRC value to the CRC values of the originals.

A file doesn't have to change much for its CRC value to be different. In fact, if even just one bit in a file changes, the CRC value for that file will change as well. If all you did was change one letter in a readme.txt file between version 1 and version 2, the CRC value for that readme.txt file would be completely different.

CRC values can be calculated for any type of file.

More on CRC values:
The basic idea of the CRC algorithm is to treat all the bits in a file as one big binary number, and then divide that number by a standard value. The remainder from the division is the CRC value.

You can think of this value as being like a fingerprint for each file. Unlike human fingerprints, however, it isn't impossible for two files to have the same CRC-32 value. Setup Factory uses an industry-standard CRC-32 algorithm which generates CRC values that are 32 bits in length. This means that one in every 4,294,967,296 files could have the same CRC "fingerprint."

Although the chances of any two files having the same CRC value are incredibly small, the CRC value alone isn't enough to guarantee an accurate identification. If you need to be *absolutely* sure, check the CRC in addition to other information about the file, such as the size of the file in bytes and its location on the user's system.

The Registry

The Registry is a central database provided by the Windows operating system where system and software configuration details are stored. It's arranged into a hierarchical structure that is similar to the way folders and files are organized on a hard drive.

The Registry is organized into six main folders, called *main keys*. Each of these main keys can contain any number of sub-folders, called *sub keys*. Information in the Registry is stored inside the main keys and sub keys as *values*.

There are three different types of values that can be stored in the Registry: String, Binary, and DWORD. String values contain strings of characters, Binary values contain binary data, and each DWORD value contains a single 32-bit ("double word") value.

Each main key in the Registry contains a different kind of information.

The six main keys are:

HKEY_CLASSES_ROOT

HKEY_CLASSES_ROOT is used to store information on different aspects of shell integration, like file associations, OLE, DDE, and drag-and-drop operations.

This key is actually a link to HKEY_LOCAL_MACHINE\SOFTWARE\Classes. Storing information in HKEY_CLASSES_ROOT is the same as storing information in HKEY_LOCAL_MACHINE\SOFTWARE\Classes, and vice-versa.

HKEY_CURRENT_USER

Like HKEY_CLASSES_ROOT, this is actually a link to another key. In this case, it points to the key in HKEY_USERS that belongs to the user who is currently logged onto the system. This is where configuration information for the current user is stored.

HKEY_LOCAL_MACHINE

HKEY_LOCAL_MACHINE contains information about system hardware, peripherals, installed software, OLE and software configuration, and other Windows configuration details.

HKEY_USERS

This is where user-specific configuration information is stored. Individual sub keys hold settings for each user, and the default settings are kept in a sub key named .Default.

HKEY_CURRENT_CONFIG

HKEY_CURRENT_CONFIG stores information for plug-and-play devices and the various hardware configurations that have been defined.

HKEY_DYN_DATA

HKEY_DYN_DATA is used by Windows to store dynamic information that changes frequently during the normal operation of Windows. It's usually best to leave this key alone.

TIP

You can read and write to the Registry by using the Read from Registry and Modify Registry actions.

File Associations

File associations tell Windows what actions can be performed on different types of files, what programs should be used to carry out each of those actions, and the paths to where those programs can be found.

Each file association associates a specific file extension with a command to use a program to do something to that file. For example, a file association can tell Windows that when files ending with `.doc` are double-clicked, they should be opened with Microsoft Word.

The program that is associated with a file extension is known as the *default viewer* for that type of file. For instance, Notepad is the default viewer for `.txt` files when Windows is installed.

File associations are stored in the Registry under HKEY_CLASSES_ROOT.

Actions

Actions are an important part of Setup Factory 6.0. They allow your installer to take care of any extra installation requirements that go beyond simply installing files. Actions can take care of everything from commonplace tasks like manipulating the Registry, launching programs, and backing up files, to exotic tasks like uploading data to web forms, manipulating strings, and downloading files off the Internet.

Actions also allow your installer to react to different situations in different ways. Does the user already have the evaluation version of your software installed? Is an Internet connection available? You can use actions to answer these kinds of questions and have your installer respond accordingly.

Each action is a specialized command that your installer can perform at run time. You can combine different actions together to form easily-managed *action lists*. Programmer-friendly actions like IF and WHILE provide advanced features like conditional execution and nested loops, without sacrificing the ease of use that Setup Factory is famous for.

Adding actions is both flexible and straightforward. On several dialogs throughout Setup Factory, there are *action tabs* that correspond to the different times during the installation process when actions can be performed. Each of these action tabs is a separate canvas upon which you can create a list of actions. The actions that you add to an action tab are performed in sequence, like lines in a program.

Actions can be performed at various stages during the installation process:

- at installer startup
- immediately before displaying a screen
- immediately after displaying a screen
- before installing all the files
- after installing all the files
- at installer shutdown

Actions can also be performed before and after the uninstallation process.

NOTE

Actions have replaced all the functionality of the *System Editors*, *Shell Operations*, and *Variables* dialogs from Setup Factory 5.0—and more.

SEE ALSO

? For more information on actions, see page 159.

Variables

Variables are special named "containers" for values that change. (The word "variable" comes from the changing nature of the values that variables represent.)

We say that values are "assigned to" or "stored in" variables. If you picture a variable as a container that can hold a value, assigning a value to a variable is like "placing" that value into a container. You can change this value at any time by assigning a new value to the variable; the new value simply replaces the old one. This ability to hold changeable information is what makes variables so useful.

Variable names in Setup Factory always begin and end with a percentage sign. At design time these variable names serve as placeholders, marking the places where the values will go once those values become known.

Wherever you use a variable like %CompanyName% in Setup Factory, the user will see the value that was assigned to that variable instead. (In this case, the user would see the value you specified for your company name on the Product Info tab of the *General Design* screen, i.e. some text like "Foobar Widgets and Gadgets Corp.")

NOTE

Normally, the user never sees the variable names—just the values they represent. (The exception is when a variable gets used before a value is assigned to it. In that case, the name of the variable is shown where the value would have appeared.)

There are two kinds of variables in Setup Factory: built-in variables, and custom variables.

Built-in Variables

Built-in variables are automatically provided for you by Setup Factory. They are used to represent common values that might differ between systems, like the location of the user's temp directory (%TempDir%) or the width of the user's display screen in pixels (%ScreenWidth%). These variables serve as pre-defined constants that you can use in the paths and conditional expressions within your Setup Factory project.

There are also built-in variables for things like your product name (%ProductName%) and copyright notice (%Copyright%). These variables are automatically defined to match the information you provide in the various fields on the Product Info tab of the *General Design* dialog. You only have to change these values in one place (in this case, on the Product Info tab), and wherever those variables are used in your setup, the new values will be shown automatically.

Finally, there are built-in variables you can use in action lists to get information about the last action that was performed (%LastCommand%), and in the case of an error, to get information about the type of error that occurred (%LastErrorNum%, %LastErrorMsg% and %LastErrorDetails%).

SEE ALSO

? For a complete list of built-in variables, see *Built-in Variables* in the Command Reference, or see page 269 of this User's Guide.

Custom Variables

Custom variables are not built into Setup Factory—instead, they're created whenever you specify a new variable name to receive a value.

Custom variables can be used to receive user input, such as the user's email address or phone number. They can also be used to receive information from the Registry or INI files, such as the path where a given software component is installed on the user's system. For instance, you would use a custom variable when you query the Registry for information that was put there by another application.

You can even make the way your installer operates depend on the values of specific variables. Using variables, you could install different files if a certain registry key exists, or skip a screen based on the user's input on the screen before.

You can also assign values to custom variables and use them as constants. For instance, you could set a variable like %FaxNumber% to your company's fax number, and then use that variable in messages throughout your installation instead of the number itself.

SEE ALSO

? For information on how to define custom variables in Setup Factory, see page 214.

Design-time Constants

Design-time constants are similar to variables, but instead of being converted to values at *run* time, design-time constants get converted at *build* time. We call them *design*-time constants because the names you give them only exist at design time. At build time, the name of each design-time constant is replaced by the value that was assigned to it.

Just like variables, you can use design-time constants as placeholders to represent values in your project. Unlike variables, however, design-time constants are replaced by their values *before* the setup executable is built. (Variable names are only replaced by their values when the installer is run.)

Essentially, design-time constants let you substitute one string for another during design time. They give you the ability to "name" a string, and use that name everywhere you want the string to appear. For example, you could assign "hello world" to a constant named #GREETING#, and include #GREETING# on several screens throughout your project. Before the installer is built, every occurrence of #GREETING# in the project will be replaced by the words "hello world."

NOTE

Each design-time constant is like one big "search and replace" operation that gets performed on your project before the setup executable is built.

Because design-time constants are in effect at build time, you can use them inside build-time conditions. Build-time conditions let you control what files get included in your setup executable. (You can't use variables in build-time conditions, because variables don't receive their values until run time.)

TIP

Make your design-time constants easy to recognize by putting a number sign or "hash" sign (#) at the beginning and end of their names.

It's a good idea to give design-time constants names that distinguish them from their variable cousins. We recommend using all-uppercase letters for your constants, and putting a number sign or "hash" sign (#) at the beginning and end of every constant's name. For example, a design-time constant for the path to your source files could be named #LOCAL_PATH#.

You can use design-time constants anywhere that you can enter text in Setup Factory.

NOTE

Design-time constants are like the C/C++ preprocessor directive *#define*. They provide you with a way to set up "aliases" or "macros" that will be replaced at build time with whatever text you specify.

Expressions

Expressions are like miniature program statements that you can use to perform calculations and tests in Setup Factory. They consist of strings containing operators and values arranged in an order or "syntax" that Setup Factory understands. Setup Factory evaluates or "interprets" these expressions and resolves them to a single result.

You can use expressions in build-time conditions, run-time conditions, screen conditions, and IF, WHILE and Assign Value actions.

Conditional Expressions

Conditional expressions are just like other expressions, but their results are interpreted as Boolean (true/false) values. In other words, the result of a conditional expression is always either true or false.

You can use conditional expressions in build-time conditions, run-time conditions, screen conditions, and IF and WHILE actions.

Boolean Values (True and False)

Boolean values are used to describe logical truths—whether something is "true" or "false." In fact, true and false are the only two possible Boolean values.

Boolean values are often used to describe the results of logical comparisons like "10 > 5" and "tree = dog." We say that "10 > 5" is *true*, because 10 is greater than 5. We say that "tree = dog" is *false*, because the string "tree" is not equal to the string "dog."

In Setup Factory, an expression is considered *true* if it resolves to either the word "true" or any non-zero integer value (such as "1" or "41395"). An expression is considered *false* if it resolves to anything else—in other words, an expression is false if it resolves to the number 0 or any string other than "true" (such as "false" or "raspberry").

NOTE

True and false are also used symbolically to represent yes/no and on/off in Setup Factory. For instance, when a check box is checked, its variable is set to True, and when it's unchecked, its variable is set to False.

Boolean values are named after George Boole (1815-1864), an English mathematician who developed the system of logical thought that came to be known as Boolean algebra. Boole's texts on symbolic logic are still used to teach mathematics, computer science and artificial intelligence today.

Build-time Conditions

Build-time conditions let you conditionally include or exclude files from your setup executable.

Every file in a Setup Factory project can have a build-time condition attached to it. The build-time condition determines whether that file gets included when the setup executable is built.

A build-time condition is a conditional expression that resolves to either true or false. If the expression resolves to true, the file gets included in the installer. If it resolves to false, the file is excluded from the installer.

By using design-time constants as switches, you can set up build-time conditions to conditionally include or exclude files from different "builds" of your installer. This enables you to easily manage concurrent versions of an installer within a single Setup Factory project.

NOTE

You can use design-time constants as switches (or "flags") by assigning different values to them before initiating the build process.

For example, you could use the same Setup Factory project to build installers for the commercial and evaluation versions of your software. Simply define a design-time constant to use as a switch—for the sake of discussion, let's name it #BUILD#—and use that constant in build-time conditions. Give the files that are only meant for your commercial release the condition #BUILD# = "full", and give the files that only belong to your demo the condition #BUILD# = "demo". This lets you control whether the files from the commercial or demo versions of your installer get included by assigning either "full" or "demo" to #BUILD#. Remember how design-time constants are replaced at build time by the values currently assigned to them? When you assign "full" to #BUILD#, the condition for the commercial files will be true, because the expression "full" = "full" is true, and the condition for the demo files will be false, because the expression "full" = "demo" is false.

NOTE

Only design-time constants can be used as flags in build-time conditions. You can't use variables in build-time conditions, because variables don't receive their values until run time.

Run-time Conditions

Run-time conditions let you control whether specific files get installed from your setup executable onto the user's system.

Every file in a Setup Factory project can have a run-time condition attached to it. The run-time condition determines whether that file gets installed when the setup executable is run.

A run-time condition is a conditional expression that resolves to either true or false. If the expression resolves to true, the file is installed. If it resolves to false, the file is not installed.

NOTE

You can also control whether files are installed by assigning them to packages. Files belonging to a package are only installed if the user selects that package at run time. However, even when a package is selected, each file's run-time condition must still be satisfied in order for that file to be installed.

Packages

Packages are special categories that you can define in order to group related files together. They're often used as part of a "custom" installation option, presenting optional components and letting users choose which parts of an application they want to install on their system.

You can have as many packages in a Setup Factory project as you want. Each package has a name and an optional description that you can use to provide information about the package to your users. This information will appear on the *Select Packages* screens that

you can include in your installer. The *Select Packages* screens allow your users to select which packages they want to install.

Each file in a Setup Factory project can be assigned to one or more packages. When a file is assigned to a package, it will only be installed if the user selects that package at run time. In other words, when users choose a package, they automatically enable the installation of all the files that belong to it.

NOTE

A file that is assigned to a package is said to "belong" to that package.

Package Variables

Each package has a unique custom variable that can be set to either true or false. When the user chooses to enable a package, that package's variable is set to "true". If the user chooses not to enable a package, that package's variable is set to "false".

When you add a *Select Packages* or *Select Install Type* screen to your installer, you're really just giving the user an easy way to set the package variables to true or false.

These true or false values are what determine whether the files assigned to each package are installed. When a package variable is set to "true", any files associated with that package *will* be installed. If a package variable is set to "false", the files associated with that package *won't* be installed, unless they also belong to another package whose variable is set to true.

NOTE

Files assigned to a single package are only installed if that package's variable is set to "true". Files assigned to multiple packages are installed if *any* of the packages' variables are set to "true".

Primer Files

Primer files are just regular files that get extracted from the setup executable *before* the installation process begins. This means you can use primer files at the very start of the installation process, right after the user runs the setup executable.

You can make any file a primer file simply by adding it to the Primer Files tab of the *General Design* dialog. The files on this tab will be included in the setup executable when you build the installer.

Primer files are automatically extracted to a temporary folder at run time. The path to this folder is stored in a built-in variable named %TempLaunchDir%. You should use this variable in actions when you need to access your primer files.

Primer files make it easy to run a program on the user's system before the rest of your files are installed. Just add the program to the list of primer files, and execute it with an Execute File action early in the installation process.

For example, you might need to execute a custom program or DLL function before your software is installed—perhaps to perform some product-specific, low-level pre-installation tests on the user's hardware, and write the results to the Registry. By adding your custom program or `.dll` to the list of primer files, you could distribute it "inside" your setup executable and still be able to run it before any of the "normal" files in your project are installed.

TIP

Another way to access files at the start of the installation process is to distribute them "alongside" your installer. For instance, you could store the files "externally" on the same CD-ROM, and access them directly; or, you could download the files from your web site using a Download File HTTP action on the Startup action tab.

Primer files are only required when you need early access, *and* you want the files to be included in your setup executable.

Chapter 3

Getting Started

The first step in creating a successful software installation is to plan it out.

In fact, planning your installation is one of the most *important* steps in designing a professional installer. Knowing what your installer needs to accomplish in advance will give you a clear goal to aim for and a solid plan to follow.

TIP

As with most things, the more planning you do now, the less repairing you'll have to do later. Investing some time in planning at the start can save you a lot of time in the long run.

There are several things you need to know in order to create an installer:

- What files do you need to distribute?
- Where do your files need to be installed?
- What system changes need to be made?
- What information do you need from the user?

This chapter will provide you with the information you need to answer these questions as quickly and accurately as possible.

What Files Do You Need to Distribute?

The first step in preparing your installation is to determine exactly what files your software requires for proper operation.

There are four basic types of files that you may need to distribute: program files, configuration files, operating system components, and shared application resources.

Program Files

Program files are the "main files" of your application. These files are essential to your application and are only useful in the context of your application. They can be executables, help files, documents, templates, or any other data files that your application requires. Program files usually make up the bulk of your software.

Configuration Files

Configuration files are used to store startup options, user settings, and other configuration options for your software. Information can be read from and written to these files during the normal operation of your application. These files are often referred to as "config" files, and come in many different formats, from traditional "INI" files (which adopt the same internal structure as Windows `.ini` files) to modern XML files.

Operating System Components

These files are usually included with the Windows operating system, but you may want to distribute the newest versions of certain files, or you may have developed custom system files of your own. These files are generally DLLs, OCX components, or hardware drivers like SYS files and VxDs.

Shared Application Resources

These are files that may be shared by more than one application, such as ActiveX controls, OCX components, and DLL files.

Some of the files that you need to distribute will be obvious, such as the main executable and help files. Others may be less apparent, such as DLLs and ActiveX controls installed in the Windows directories of your development system.

NOTE

Many of today's development tools require that you distribute runtime support files along with your application. Please consult your development tool's documentation to determine what files you need to distribute with your software.

Often an executable in your application absolutely requires other files in order to work properly. These "absolutely required" files are known as *dependency files*.

Dependency Files

Dependency files are external support files that an executable requires for proper operation. In other words, they are external files that a program file "depends on" in order to function properly. Dependency files may include INI files, DLLs, ActiveX controls, OCX components, or any other support file type. Although it's generally preferable to install dependency files in the same directory as the program that needs them, they are often installed in other locations, such as the Windows system directories.

SEE ALSO

? Setup Factory has a built-in dependency scanner that you can use to identify dependency files and add them to your project. For more information on this feature, see page 203.

Setup Factory also provides built-in runtime support for many runtime dependency files via the *Runtime Support* dialog. The *Runtime Support* dialog makes it easy to add support for such technologies as Visual Basic, ODBC, DAO, ADO, and many more. For more information on this feature, see page 199.

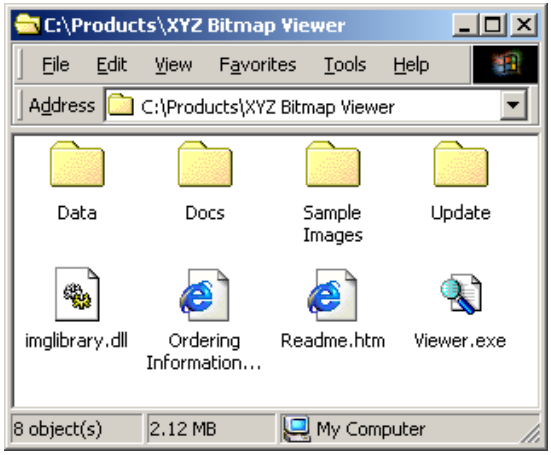
Preparing the Directory Structure

The ultimate goal of a good installer is to get your software onto the user's system in an easy and accurate manner. Although Setup Factory ensures both you and your users' ease of use, the accuracy of the installation itself is largely up to you. Whether your files are installed in a structure in which they can function properly depends largely on where you tell Setup Factory to install those files.

The best way to ensure an accurate installation is to prepare the software in its finished state on your development system before you begin creating the installer. This means setting up the entire directory structure and all of the file locations exactly as you want them to appear on the user's system.

The end result should be that your software is fully installed on your development system exactly as you want it installed on the user's.

Not only does this make it easy to test your software in its intended directory structure, but it also makes the process of designing the installer much quicker and easier for you. All you'll have to do is drag your application folder onto the Setup Factory project window, and the files and sub-folders will be created exactly the same on the user's system.



A well-prepared directory structure

TIP

You should fully test your software in its "final" structure before creating the installer.

Where Do Your Files Need To Be Installed?

Once you have your software organized, you need to determine exactly where each file needs to go on the user's system. Although Setup Factory does a lot of this work for you by maintaining directory structures when you add files to your project, there are still some files that may need to be directed to different locations on the user's system.

Here are some guidelines to help you determine where you should install your files on the user's system. Once again, there are four basic types of files that you may need to

distribute: program files, configuration files, operating system components, and shared application resources.

Installing Program Files

Program files should be installed in a directory that the user chooses during the installation process. Throughout this manual and in the Setup Factory program, this directory is referred to as the application directory. The path to the application directory is represented by the built-in variable %AppDir%.

SEE ALSO

? For more information on built-in variables, see pages 36 and 211.

It's okay to install program files in sub-folders within the application directory—in fact, organizing your program files into sub-folders is a very good idea. Your files don't all have to be in the application directory itself; the folder that the user chooses can be used as a common application directory, with sub-folders for all of your program files.

If the users aren't given an opportunity to choose an installation folder, the path that you provided as the default value for %AppDir% will be used. You can provide a default path for the application directory on the Settings tab of the *General Design* dialog.

SEE ALSO

? For more information on the Settings tab, see page 129.

Installing Configuration Files

In the past, initialization or "INI" files were often installed in the Windows directory (%WinDir%). This is definitely not necessary or even beneficial in all cases. Unless your application shares a configuration file with other programs, it's best to install the file in the application directory along with your program files.

You should avoid installing any files in the Windows directory, or in any other folders where critical operating system files are stored, unless it is absolutely required by your application.

Installing Operating System Components

Operating system components, such as DLL or OCX files, are traditionally installed in the WINDOWS\SYSTEM directory (%SysDir%). If your application doesn't need to share these files with other applications, it's best to install them in your application directory (%AppDir%) instead.

SEE ALSO

? For a complete list of built-in variables like %SysDir%, %WinDir%, and %AppDir%, see *Built-in Variables* in the Command Reference or page 269 in this User's Guide.

Installing Shared Application Resources

Shared application resources, such as some ActiveX controls or DLL files, are generally installed in the WINDOWS\SYSTEM directory (%SysDir%). Keep in mind that many DLL and OCX files are COM servers, also known as "OLE components" and "ActiveX controls." As such, they will need to be registered with the operating system before they will be available for use by your application.

Setup Factory will automatically try to detect files that require registration when you add them to your project. You can also manually indicate files that you want Setup Factory to register by using the options on the Advanced tab of the *File Properties* dialog. And, you can use a Register File action to register a file manually at run time.

SEE ALSO

? For information on how to use the *Preferences* dialog to control whether Setup Factory will automatically try to register DLL and OCX files, see "Setting Automatic File Treatment Options" on page 62.

For more information on how to manually indicate files that you want Setup Factory to register, see page 125.

For more information on the Register File action, please consult the Command Reference.

Keep in mind that some DLL and OCX files have dependency files themselves, and these dependency files need to be distributed and possibly also registered before the DLL or OCX files can be used. You should consult the documentation for your components to determine what dependencies they have.

NOTE

Many OCX and DLL controls ship with a dependency file. The dependency file typically has the same filename as the control with a .DEP extension, e.g., Control.ocx would have a dependency file named Control.dep. These dependency files can be opened and viewed with a text editor (such as Notepad) and can tell you a lot about what, if any, dependencies the control may have.

SEE ALSO

? You can also use Setup Factory's built-in dependency scanner to identify any dependency files your components may have. For more information on this feature, see page 203.

What System Changes Need To Be Made?

The next step is to think about what, if any, changes need to be made to the user's system. This can include changes to system files (such as WIN.INI, CONFIG.SYS, SYSTEM.INI and AUTOEXEC.BAT), changes to the Registry, and even such things as installing and registering fonts. All of these changes can be handled easily using Setup Factory actions.

SEE ALSO

? For more information on actions, see pages 34 and 159.

What Information Do You Need from the User?

The final step in planning your software installation is to consider what information you will need from the user, and what information you will need from the user's system. For example, in order to personalize your software or register it to a specific user, you may want to ask the user to provide their name on a *User Information* screen. Or, you may want to ask the user for a serial number on a *Verify Serial Number* screen.

SEE ALSO

? For more information on screens, see page 141.

If you need to gather information from the user's system, you might be able to use a built-in variable. For example, if you want to know the user's company name, you could try using the built-in variable %RegOrganization%.

Otherwise, you can use one of several sleuth-worthy Setup Factory actions to query the Registry or investigate any other part of the user's system that the installer has access to.

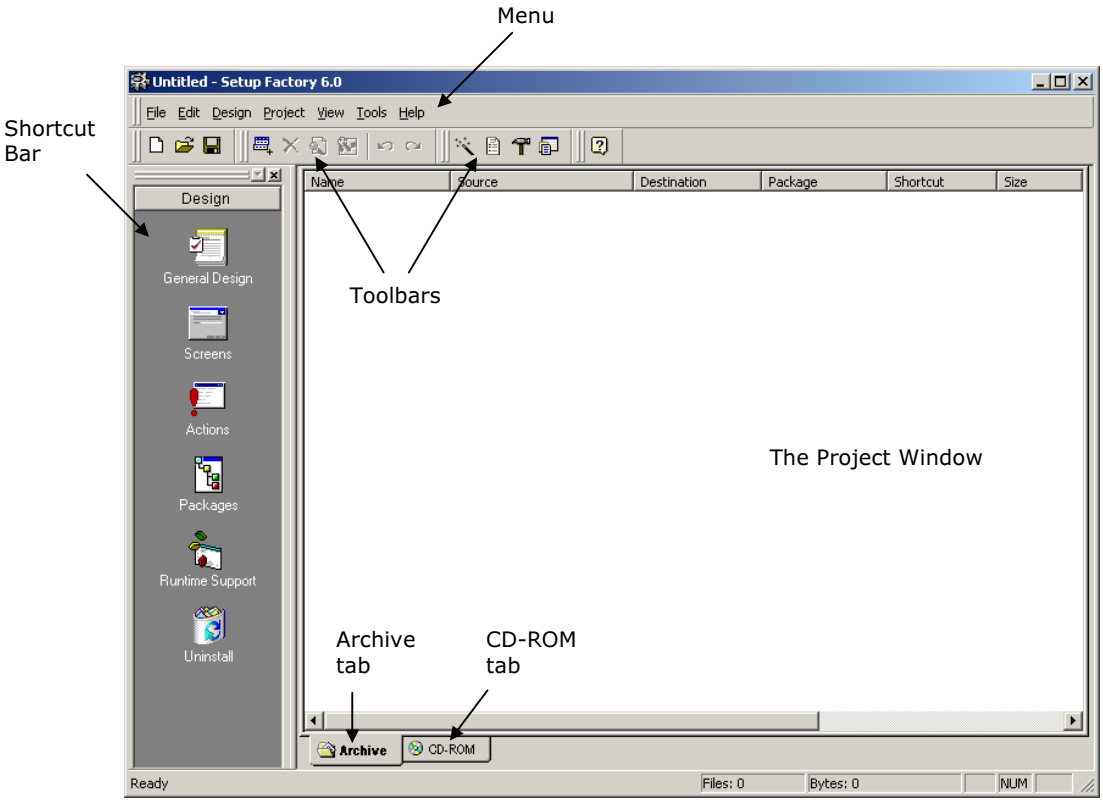
SEE ALSO

? For a complete list of actions, see *Actions Index* in the Command Reference, or see page 263 in this User's Guide.

Chapter 4

The Design Environment

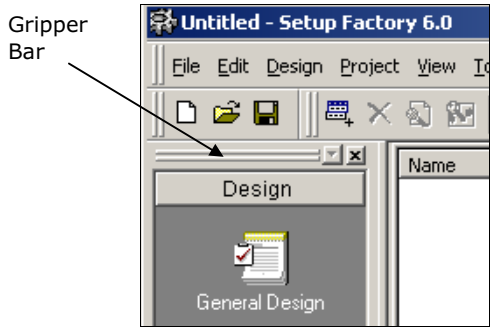
This chapter will help you get familiar with the Setup Factory design environment as quickly as possible. It will provide you with a quick tour of the Setup Factory interface, and help familiarize you with some of the terms used throughout this user's guide.



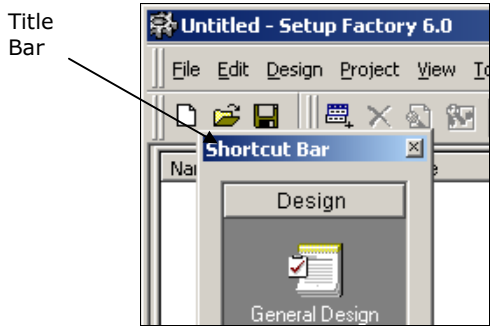
The Setup Factory main screen

Shortcut Bar

The shortcut bar is used to quickly access the Setup Factory design dialogs. It's located along the left side of the main screen by default, but you can dock it along the right side or let it float above the main screen if you prefer. To move the shortcut bar when it's docked, left-click on the gripper bar at the top, hold the mouse button down, and drag the shortcut bar to the desired location.



To move the shortcut bar when it's floating, drag it by the title bar instead.



To return the shortcut bar to its docked position, drag it to the left or right edge of the Setup Factory main screen.

You can toggle the shortcut bar on or off at any time by selecting **View | Shortcut Bar** from the menu.

There are six design dialogs you can access from the shortcut bar:

General Design

The *General Design* dialog is used to configure your product information, languages, serial numbers, and other important installation settings.

Screens

The *Screens* dialog allows you to configure the screens that will be displayed by your installer at run time.

Actions

The *Actions* dialog is used to define lists of actions that will be performed at run time. You can use actions to perform advanced installation tasks like manipulating variables, modifying the Registry, starting and stopping programs, and more.

Packages

The *Packages* dialog is used to define and localize packages. Packages are usually used to group files and provide custom installation options to users.

Runtime Support

The *Runtime Support* dialog allows you to select the runtime technologies you want to include in your project. This includes Visual Basic 5.0 and 6.0, Visual C++ 6.0, Visual FoxPro and many more.

Uninstall

The *Uninstall* dialog is where you can configure the uninstall routine for your installer.

TIP

You can also access these dialogs at any time from the **Design** menu.

Toolbars

The Setup Factory toolbars make it easy to access features quickly without having to locate a menu command. Each button on a toolbar corresponds to a menu item or command. The toolbars are fully customizable and can be positioned anywhere on the Setup Factory screen. The toolbars can either float above the main screen or be docked along the top, bottom, left or right edges.

The File and Edit toolbars

You can customize the existing toolbars or create your own by selecting **Tools | Customize Toolbar** from the menu. You can also show or hide any of the toolbars by using the *Toolbars* dialog, which you can access by selecting **View | Toolbars** from the menu.

Project Window

The project window takes up most of the main screen and contains a list of all the files that you have added to your project. From this list you can highlight specific files and view or edit their properties.

There are two tabs on the project window: the Archive tab, and the CD-ROM tab.

Archive Tab

The Archive tab is for files that you want compressed and stored in the setup executable. In other words, this tab is for files that will be included *in* your installer.

When Setup Factory builds the setup executable, it compresses all the files listed on the Archive tab and packs them right into the setup executable file. We say that these files get included in the *setup archive*, which is how the Archive tab was named.

CD-ROM Tab

The CD-ROM tab is for files that you *don't* want compressed and stored in the setup executable. In other words, this tab is for files that will be distributed *with* your installer.

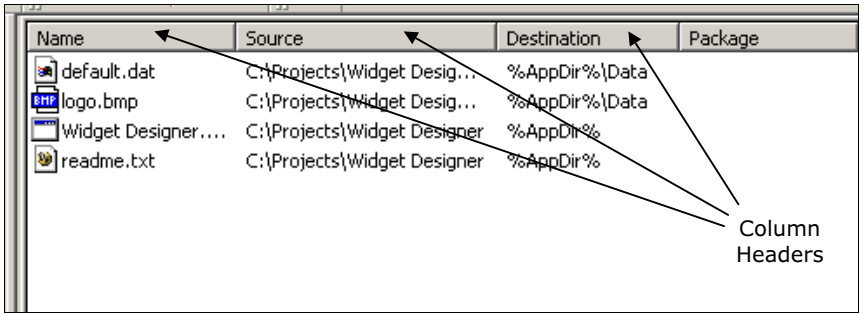
When Setup Factory builds the setup executable, it includes all the information it knows about the files on the CD-ROM tab, such as where the files are expected to be at run time, and where you want them to be installed to. But it doesn't actually include the files in the setup executable. Instead, you're expected to make sure that those files will be where the installer expects them to be at run time. Usually this is done by distributing the files along with the setup executable on the same CD-ROM (hence the name "CD-ROM tab").

NOTE

The CD-ROM tab isn't just for files that are distributed on CD-ROMs, though. You can use it to install any external files, i.e., any files that are external to the setup executable.

Column Headers

Both tabs have columns that display different information about each file in your project. You can sort the information along any of the columns by clicking on the header for that column. If you click on the same header again, the files will be sorted by that category in reverse order.



To sort files by their extensions, select "Sort filenames by type" on the General tab of the *Preferences* dialog, which you can access by selecting **Edit | Preferences** in the menu.

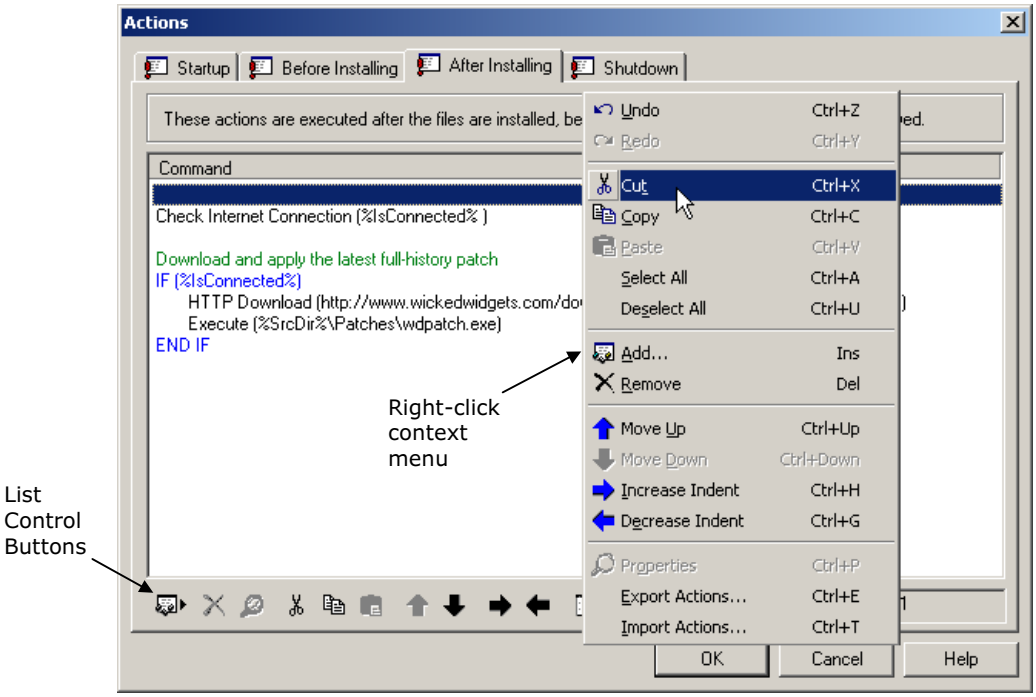
NOTE

The order of files on the project window doesn't have any bearing on the order they're installed in. Files are installed in the same order they were added to your Setup Factory project. Sorting files in the project window has no effect on the installation order at all.

Whenever you load a project, Setup Factory displays the files in the order they will be installed.

Lists

Setup Factory makes extensive use of lists in its user interface. Most of these lists have similar controls to let you add, remove, edit and rearrange items in the list.



List controls on the After Installing actions tab

There are three ways the list controls can be accessed: using buttons, using the right-click context menu, and using hotkeys.

List Control Buttons

Here are some of the common list control buttons that appear throughout Setup Factory:

	**Add**	Add a new item to the list
	**Remove**	Remove the currently selected item(s) from the list
	**Properties**	Open the Properties screen for the currently selected item
	**Cut**	Remove the selected items and place them in the clipboard
	**Copy**	Copy the selected items into the clipboard
	**Paste**	Add the items currently in the clipboard into the list
	**Move up**	Move the selected item up one position
	**Move down**	Move the selected item down one position
	**Increase indent**	Indent the selected items more
	**Decrease indent**	Indent the selected items less
	**Advanced**	Display a menu of list-specific functions like "Import Registry Values" and "Export Actions."

Right-click Context Menus

You can also access the list controls by right-clicking on a list and using the right-click context menu. Some context menus have additional controls, such as the Select All command, that can only be accessed via the menu; they don't have any buttons associated with them.

TIP

Right-click context menus are available throughout Setup Factory. You can even use them to insert items into edit fields.

Hotkeys

You can access the list controls using the keyboard by pressing certain key combinations known as *hotkeys*. The hotkeys are listed on the right-hand side of the context menu when you right-click on a list.

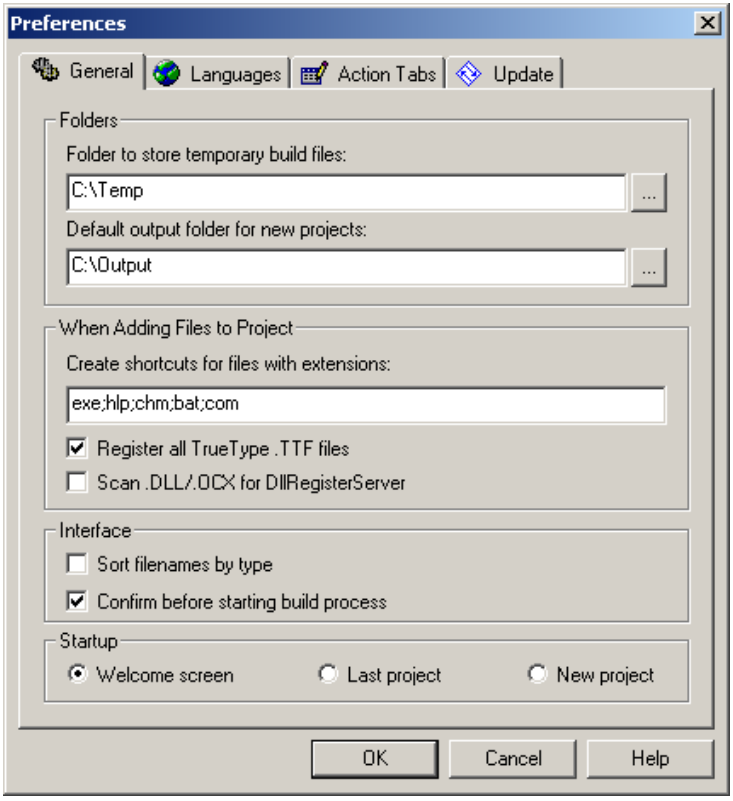
For example, the hotkey for the Select All command is Ctrl+A. This means you can select all items in a list by holding the Ctrl key and pressing the A key on your keyboard.

TIP

Hotkeys are available throughout Setup Factory. Just about every menu item has a hotkey associated with it.

General Preferences

You can set general preferences for the Setup Factory design environment on the General tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.



The General tab of the Preferences screen

Setting the Temporary Build Folder

Setup Factory uses a temporary folder while generating the setup executable. You can change the folder that is used for this purpose by editing the path in the **Folder to store temporary build files** field, which is located on the General tab of the *Preferences*

dialog. You can also press the **Browse** button () to browse for a path using the *Select Folder* dialog.

Setting the Default Output Folder

You can change the default output folder for new projects by editing the path in the **Default output folder for new projects** field, which is located on the General tab of the *Preferences* dialog. You can also press the **Browse** button () to browse for a path using the *Select Folder* dialog.

Setting Automatic File Treatment Options

When you add certain types of files to your project, Setup Factory will automatically configure their file properties according to the settings in the **When Adding Files to Project** section on the General tab of the *Preferences* dialog.

There are three different automatic file treatment options that you can control on this screen:

The **Create shortcuts for files with extensions** field determines which types of files will automatically have the **Create shortcut in Start menu** option enabled.

The **Register all TrueType .TTF files** check box determines whether TrueType font files will automatically have the **Register as TrueType font** option enabled.

The **Scan .DLL/.OCX for DllRegisterServer** check box determines whether DLL and OCX files will be scanned to see if they support registration—i.e., whether they have the DllRegisterServer function available. If a file supports registration, the **DllRegisterServer** option will be enabled for that file on the Advanced tab of the *File Properties* dialog.

Sorting Filenames by their Extensions

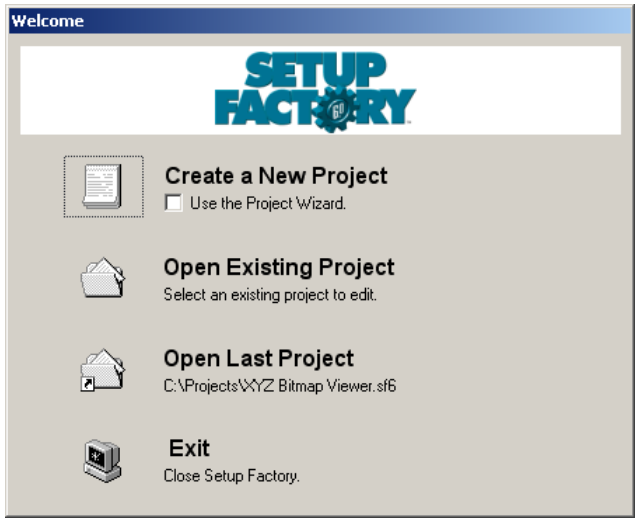
You can sort files by their extensions on the Project Window by selecting the **Sort filenames by type** check box on the General tab of the *Preferences* dialog.

Enabling or Disabling Build Process Confirmation

By default, Setup Factory will ask you to confirm that you really want to start the build process whenever you build the setup executable. You can modify this behaviour by selecting or clearing the **Confirm before starting build process** check box on the General tab of the *Preferences* dialog.

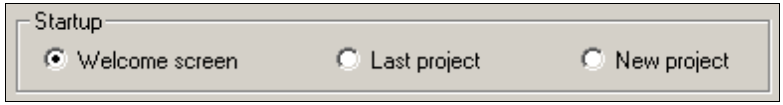
Choosing Startup Options

When you start Setup Factory, it opens a welcome screen by default to let you quickly start a project, open an existing project file, or open the project file you had open the last time you ran Setup Factory. If you prefer, you can have Setup Factory automatically create a project or open the last project file instead of displaying this welcome screen.



Setup Factory's welcome screen

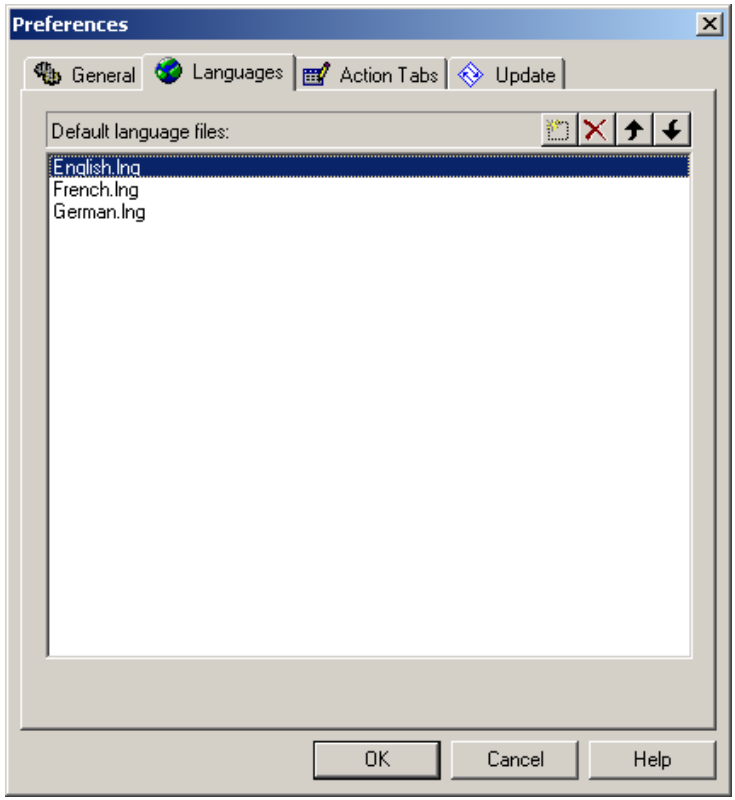
You can choose between the three options in the Startup section on the General tab of the *Preferences* dialog.



The Startup section on the General tab of the Preferences dialog

Language Preferences

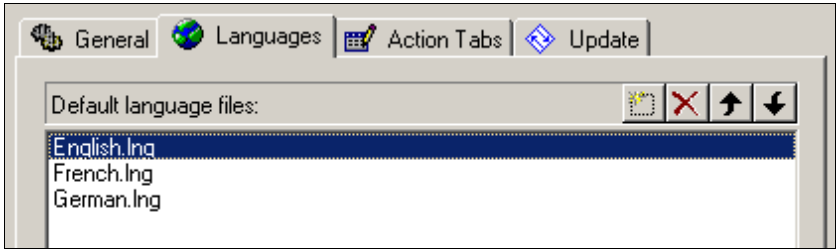
You can set language preferences for Setup Factory on the Languages tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.



The Languages tab of the Preferences dialog

Setting Default Language Files

To change the list of languages that are added to every new project, edit the **Default Language Files** list on the Languages tab of the *Preferences* dialog. The first language in this list will also be marked as the default language—in other words, when you start a new project, the default language check box on the *Edit Messages* dialog will be selected for the first language in the **Default Language Files** list.



These three languages will be added to every new project

To add a language to the list, press the **Insert Language** button ( ) or use the Insert hotkey. Then, enter the path to the language (.lng) file you want to add, or press the **Browse** button (...) to browse for a language file.

To remove a language from the list, select the language you want to remove and press the **Remove Language** button ( ) or use the Delete hotkey.

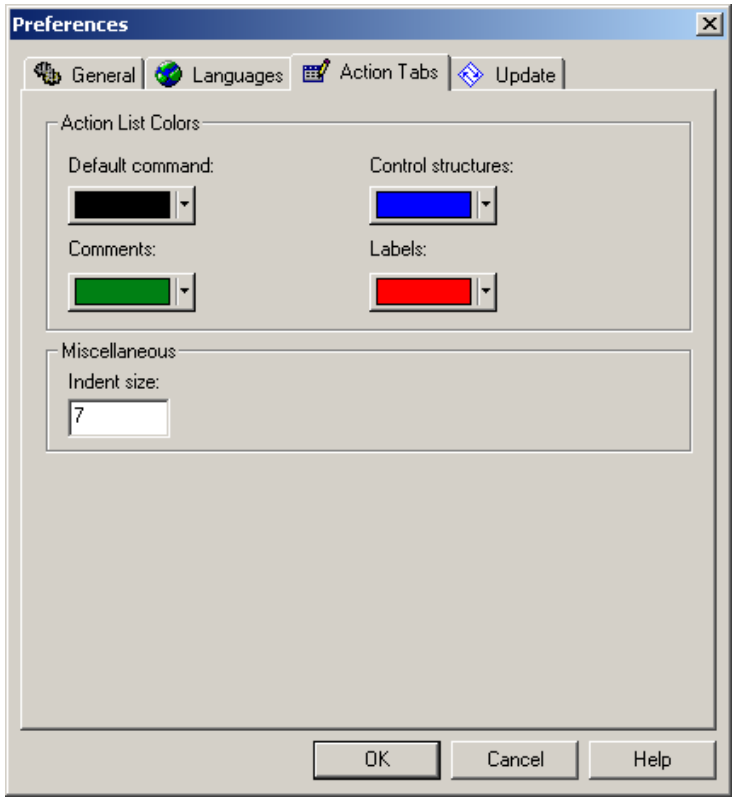
To change the order of the languages in the list, select the language that you want to move, and then use the **Move Up** ( ) and **Move Down** ( ) buttons to reposition the language in the list.

NOTE

You can add languages to the current project on the Languages tab of the *General Design* dialog.

Action Tabs Preferences

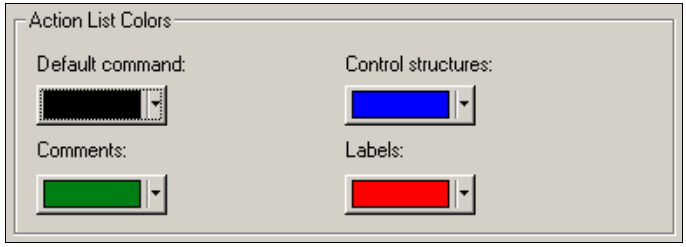
You can set preferences for all action tabs in the Setup Factory development environment on the Action Tabs tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.



The Action Tabs tab of the Preferences dialog

Changing the Action List Colors

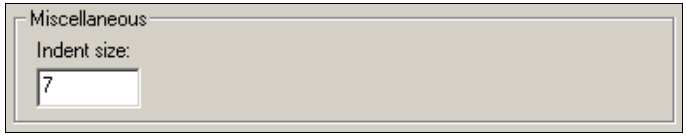
You can change the colors that are used for the control structures, comments, labels and default commands (which is basically "everything else") on the action lists. To do so, just click on one of the four color choosers on the Environment tab of the *Preferences* dialog, and select a different color from the list that appears.



The four color choosers on the Environment tab of the Preferences dialog

Changing the Indent Size

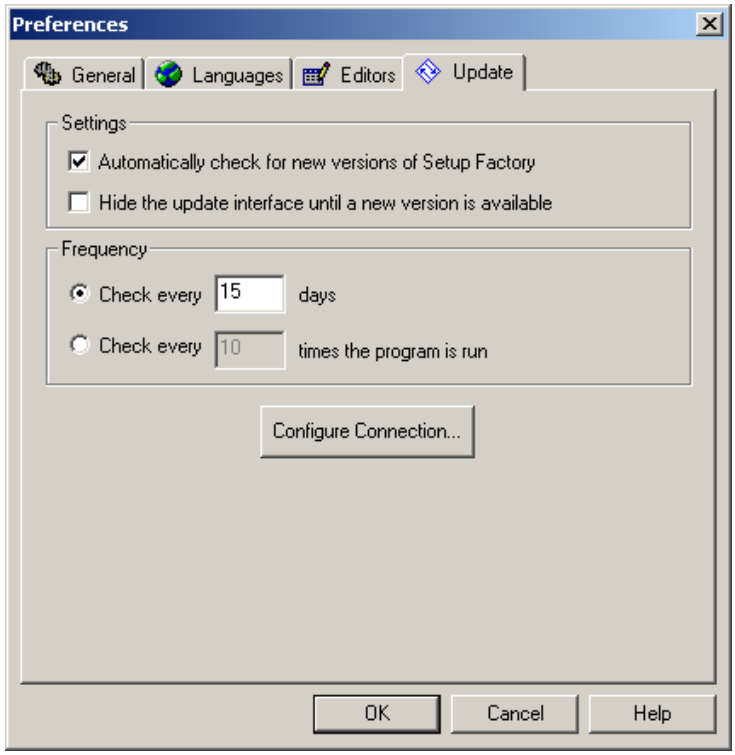
You can change the indent size used on the action tabs. To do so, enter the width in spaces for each level of indentation in the **Indent size** field on the Environment tab of the *Preferences* dialog.



The Indent size field

Update Preferences

You can set update preferences for the Setup Factory software on the Update tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.



The Update tab of the Preferences dialog

Automatically Checking for New Versions of Setup Factory

The Setup Factory development software can automatically check the Internet for updates. Setup Factory will use Indigo Rose's TrueUpdate technology to download and install new versions of itself as soon as they become available.

To enable this feature, just select the **Automatically check for new versions of Setup Factory** check box.

TIP

You can easily integrate the same update technology into your software. For more information, visit www.indigorose.com/trueupdate/.

Hiding the Update Interface Until a New Version is Available

The **Hide the update interface until a new version is available** check box lets you control whether the TrueUpdate client interface is shown each time Setup Factory checks the Internet for a new version of itself. If the check box is selected, the client interface is only shown when a new version is available and an update is performed.

Setting How Often Setup Factory Checks for Updates

The Frequency section of the Update tab allows you to specify how often Setup Factory checks the Internet for new versions of itself. You can have Setup Factory check every X number of times it is started, or have it check (on starting Setup Factory) if a minimum number of days have elapsed since the last time a check was performed.

Configuring the Setup Factory Connection Settings

You can configure the connection settings Setup Factory will use when it checks for an update by pressing the **Configure Connection** button. This will open the *Connection Settings* dialog, where you can specify whether your development system connects to the Internet using a LAN or a dial-up connection. This is also where you can configure your proxy settings if your system connects to the Internet through a proxy server.

TIP

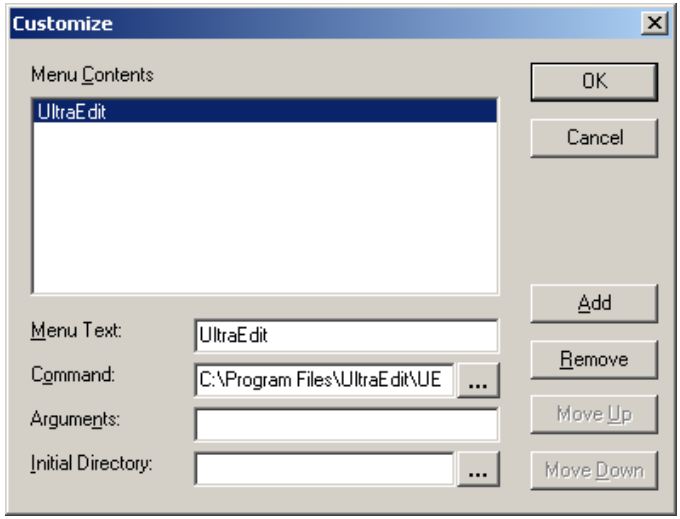
If you're development system connects to the Internet through a proxy server, you will need to configure your proxy settings in order for Setup Factory to download updates successfully.

User Tools

You can add custom menu items to Setup Factory's **Tools** menu to start other programs that you use often during the design process. For instance, you might want to add a menu item to launch your favorite text editor.

Configuring User Tools

To configure the custom items that will show up in the **Tools** menu, select **Tools | Configure User Tools** from the menu to open the *Configure User Tools* dialog.



The Configure User Tools dialog

You can use this dialog to add tools to the menu, remove them from the menu, and change the order of the tools in the menu.

TIP

It can be useful to set up a user tool to run the setup executable from within Setup Factory, so you can easily test it after it's built. Simply use the path and filename that your setup executable is normally generated to (for example, `C:\Output\setup.exe`) in the **Command** field on the *Configure User Tools* dialog.

Chapter 5

Quickstart Tutorial

This tutorial is a great place to start if you're new to Setup Factory. It will walk you through the steps required to build a simple but complete installer.

NOTE

There is definitely more than one way to approach the design process with Setup Factory. The method described in this chapter is only meant to serve as an example to get you building installers as quickly as possible.

Step 1: Prepare Your Files

As explained in the Getting Started chapter on page 47, the first and most important step in creating an installer is to prepare the directory structure and copy all the files to your development system.

Before you move on to step 2, your product should be fully installed in a neat and organized manner, looking exactly as you want it to look on the end user's system.

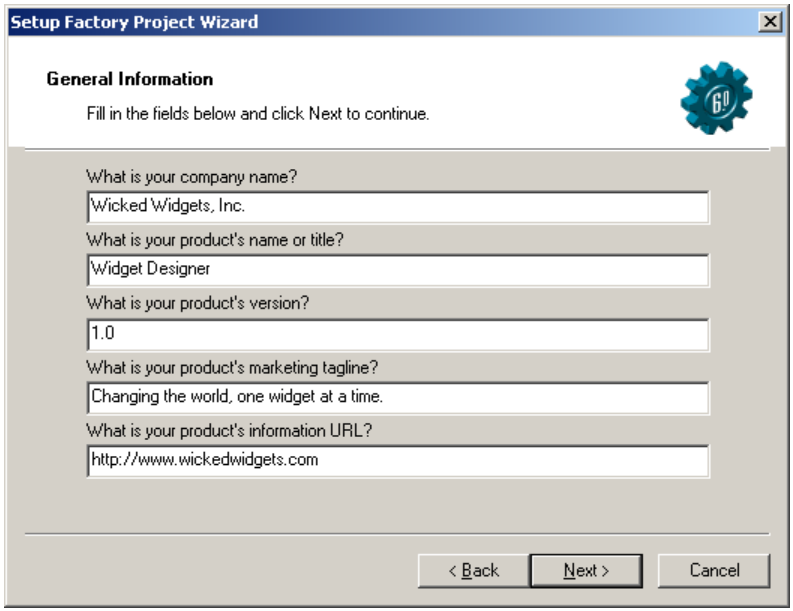
If at all possible, try to arrange it so that all of the files in your application fall within a single, common "application directory." This will make it much easier to add all the files to your project in step 2.

Step 2: Use the Project Wizard

The Project Wizard is designed to help you get a project started as quickly as possible. In some cases, the Project Wizard is all you need to create a complete installer. Of course, you will probably want to make a few modifications using the design environment—just to make everyone think you're working harder than you are...

To start the Project Wizard, select **Project | Project Wizard** from the Setup Factory program menu, or press the **Project Wizard** button ( ) on the Project toolbar. This will open the Project Wizard's *Welcome* screen.

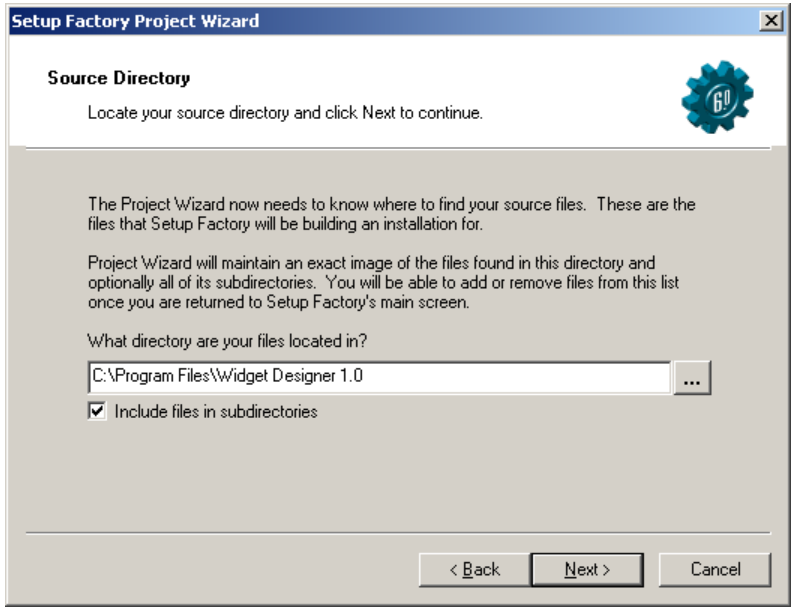
After you have read the information on the *Welcome* screen, click **Next** to continue to the *General Information* screen.



The Project Wizard's General Information screen

The *General Information* screen is used to collect information about the product and your company. This information will be used in various places throughout your installer. Fill in the information on this screen completely and accurately—it's designed to save you time later.

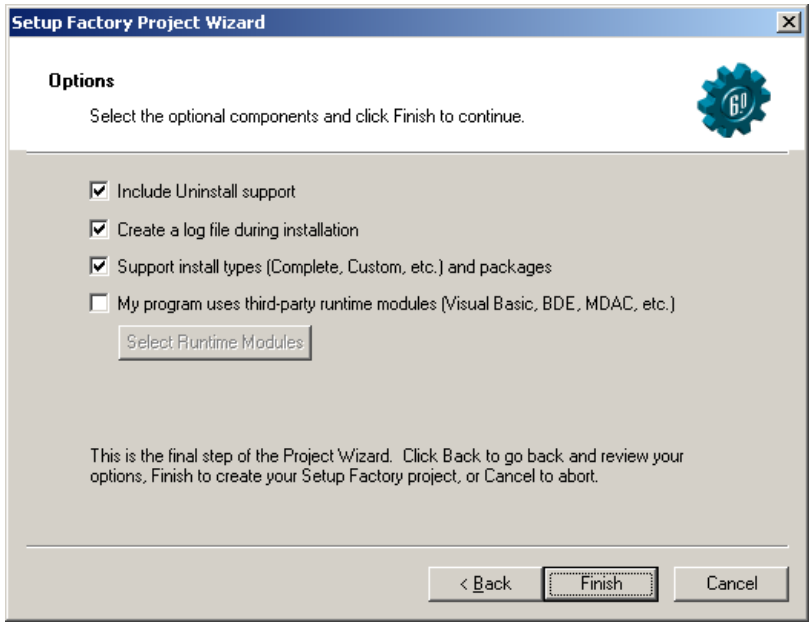
Once you've finished entering your information, click **Next** to continue to the *Source Directory* screen.



The Project Wizard's Source Directory screen

The *Source Directory* screen is used to locate your application directory. Enter the path to the top level directory where your product is installed, or press the **Browse** button (...) to browse for the directory. This is the single main directory where all of your application's files and subdirectories are found. Be sure to select the **Include files in subdirectories** check box so the Project Wizard will import all the files in your directory tree.

Once you've provided the path to your application directory, click **Next** to continue to the *Options* screen.



The Project Wizard's Options screen

The *Options* screen is the final step of the Project Wizard. You can use this screen to set some common project options, such as whether an uninstall routine will be included in the installer, whether a log of the installation process will be created on the user's system, whether a *Select Install Type* and *Select Packages* screen will be added to your project for you, and whether your product requires any third-party runtime support.

For this tutorial, select the **Support install types** option to enable support for install types and packages. Then, click **Finish** to complete the Project Wizard process.

Step 3:
Add Additional Files to the Project

Although you've added all of the main program files to your project, you may still need to add additional files that aren't found within your application directory structure. For example, you may need to add an ActiveX control that is needed by your software. Like most ActiveX controls, it would probably be located in the Windows System directory on your development system.

To add an additional file to your project:

1. Select **Edit | Add Files** from the menu.

2. Browse for the file on your system, and select the file.

3. Press the **Add** button to add the selected file to your project.

The file should now appear on the project window along with the files that were added by the Project Wizard. Note that the file's destination is automatically set to a built-in variable like %AppDir%, %SysDir%, or %WinDir%, depending on where the file was located on your development system. These variables are automatically expanded to the corresponding paths on the user's system at run time. For example, %AppDir%, %SysDir%, and %WinDir% would be expanded to the paths to your application directory, the Windows System directory, and the Windows directory on the user's system.

SEE ALSO

? For more information on built-in variables, see pages 36 and 211.

For more information on adding files to your project, see page 111.

Step 4: Create Shortcuts

If you look in the Shortcut column on the project window, you will see that Setup Factory has automatically selected some of your files as candidates for shortcuts. (In the case of our Widget Designer example, Setup Factory automatically created shortcuts for the executables and help files.) The only problem is that Setup Factory uses the filenames for the shortcut descriptions by default. In order to give your shortcuts more appropriate descriptions, you'll need to edit the properties for those files.

To change a shortcut's description:

1. Select the file whose shortcut description you want to change on the Setup Factory project window.

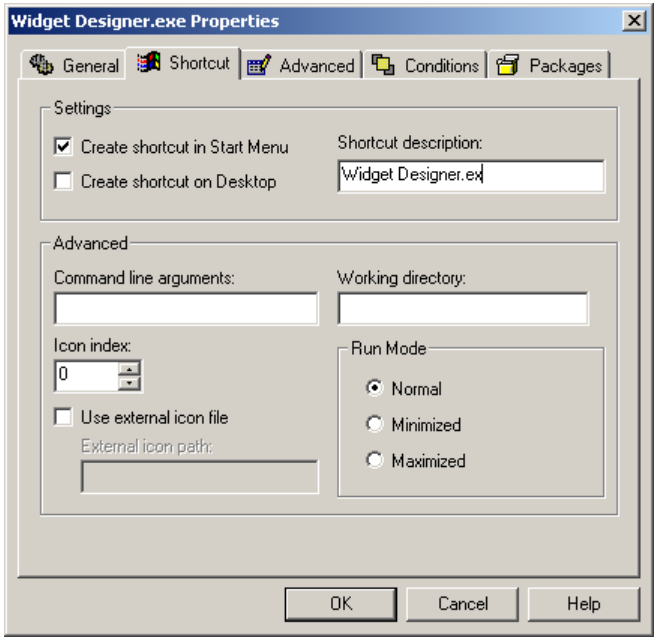
2. Select **Edit | File Properties** from the menu. This will open the *File Properties* dialog for the file you selected.

   The *File Properties* dialog is where you can edit file-specific settings, such as the source path, destination path, shortcut options, package associations, run-time and build-time conditions, and more.

3. Select the Shortcut tab.

   The Shortcut tab is where you can configure the shortcut options for the file you selected.

4. Edit the text in the **Shortcut description** field to your liking. This is the text that will appear next to the shortcut icon in the user's Start menu, and beneath the shortcut icon on the user's desktop.



Editing the shortcut description on the Shortcut tab of the File Properties dialog

5. Press **OK** to accept the changes.

You can repeat this process for any other file that you want to change the shortcut description for. You can also instruct Setup Factory to create shortcuts for other files in your project that don't have shortcuts created for them by default.

To create a shortcut in the user's Start menu for a file that does not have a shortcut created by default, such as a `.txt` file, follow the same five steps above, but enable the **Create shortcut in Start Menu** option between steps 3 and 4.

You can also enable the **Create shortcut on Desktop** option to have the installer create a shortcut on the user's desktop as well.

Before you continue, take time to explore the *File Properties* dialog and familiarize yourself with the various options available to you. If you want to learn more about any of the options, press the **Help** button ().

SEE ALSO

? For more information on shortcuts, see page 30.

For more information on the *File Properties* dialog, see page 116.

Step 5: Set Up Packages

If you have a complex or multi-product installation, you may want to offer your users several different installation options. This gives them the power to tailor the installation to meet their own specific requirements. While it may seem that this sort of flexibility would be difficult to offer, Setup Factory makes it as easy as possible.

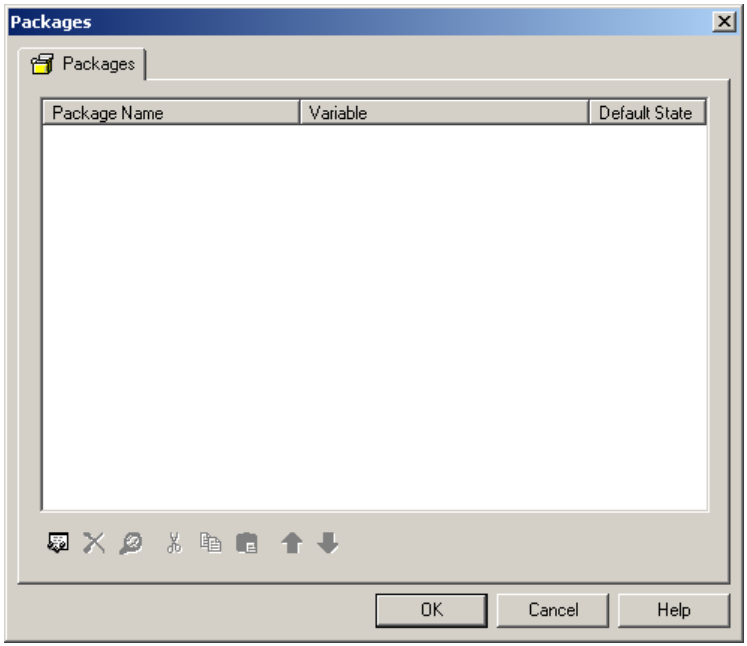
It's really simple. You start out with all the files you've added to your project. You then group your files into what we call "packages." A package is simply a related group of files, such as help files, graphic files, samples—anything you want.

Then, you add a *Select Install Type* screen and, usually, a *Select Packages* screen to your project. Your users will use these screens to select the packages that they want to install.

The first step in implementing packages is to create the packages themselves, i.e., set up the categories that your files will belong to. To do this:

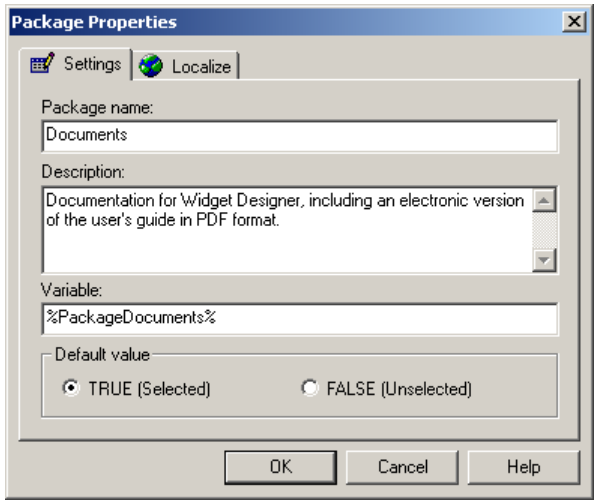
1. Select **Design | Packages** from the menu. This will open the *Packages* dialog.

   The *Packages* dialog is where you can add packages to your project, and remove or edit any packages already in your project.



The Packages dialog

2. Press the **Add Package** button ( ). This will open the *Package Properties* dialog for the new package you're adding.



The Package Properties dialog

3. Provide a name for the package in the **Package name** field. For our example, we wanted to prepare a package to group our documents together, so we named the package "Documents."

4. Provide a description for the package in the **Description** field. This is what the users will see on the *Select Packages* screen when they select this package.

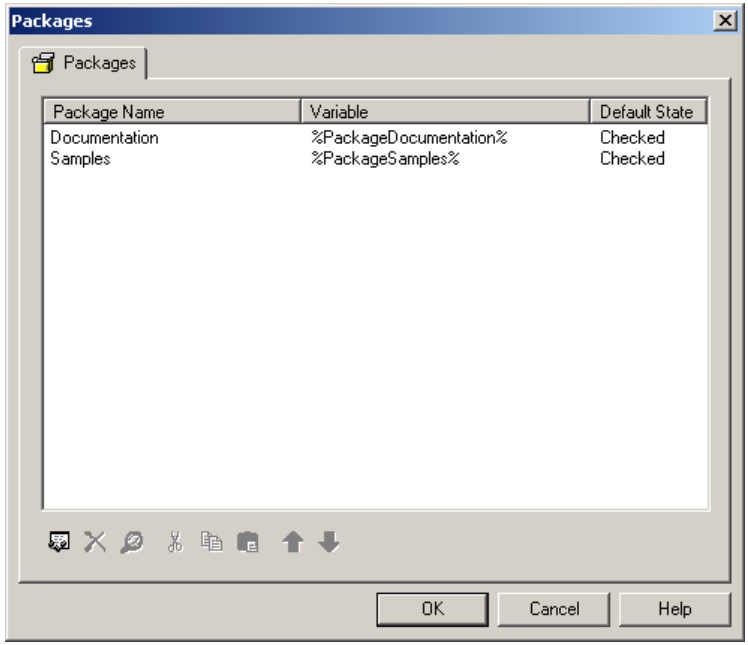
5. Enter a *unique* variable name for this package in the **Variable** field. A good way to name this variable is to start with "%Package", add the name of your package (without spaces), and end with "%". For our example, we used %PackageDocuments% as the variable name.

   Starting all your package variable names with a prefix like "Package" can help you recognize the package variables when you're working on your project.

6. Press **OK** on the *Package Properties* dialog to add the package to your project.

7. Repeat steps 2 to 6 for every package you want to add to your project.

8. Press **OK** on the *Packages* dialog to add all the packages you created above to your project. (Don't just press Cancel, or the changes you made won't be saved.)



The Packages dialog after we added 2 packages

SEE ALSO

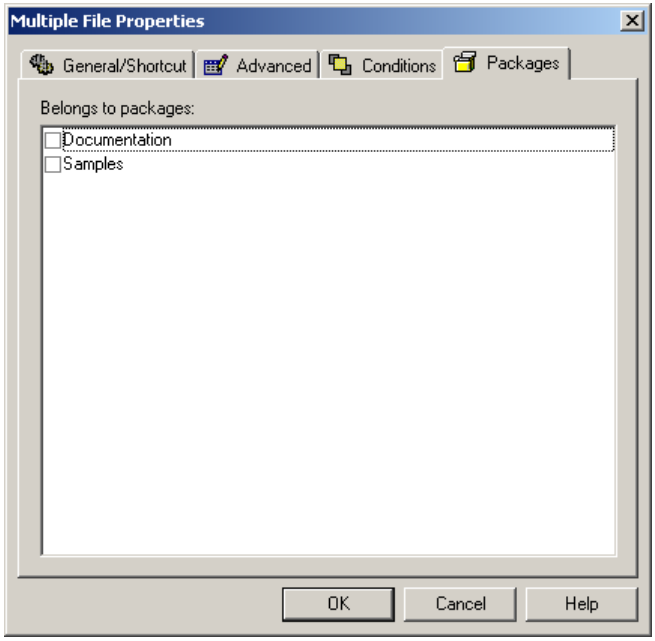
? For more information on adding packages, see page 191.

The next step in implementing packages is to assign files to the packages that you created. To do this:

1. On the project window, select the files that you want to assign to a package.

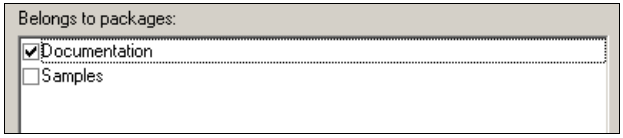
2. Select **Edit | File Properties** from the menu. This will open the *Multiple File Properties* dialog. (We'll assume you selected more than one file to add to this package. If you didn't, that's okay; the *File Properties* dialog will be opened instead, but it works pretty much the same as the *Multiple File Properties* dialog does for assigning files to packages.)

3. Select the Packages tab. This is where you can assign the files you selected to the packages you created.



The Packages tab of the Multiple File Properties dialog

4. Select the packages that you want the files you selected to belong to. The files will belong to a package when there is a solid black check mark next to that package name on the Packages tab.



After assigning our files to a package

5. Press the **OK** button to assign the files to the packages you selected.

SEE ALSO

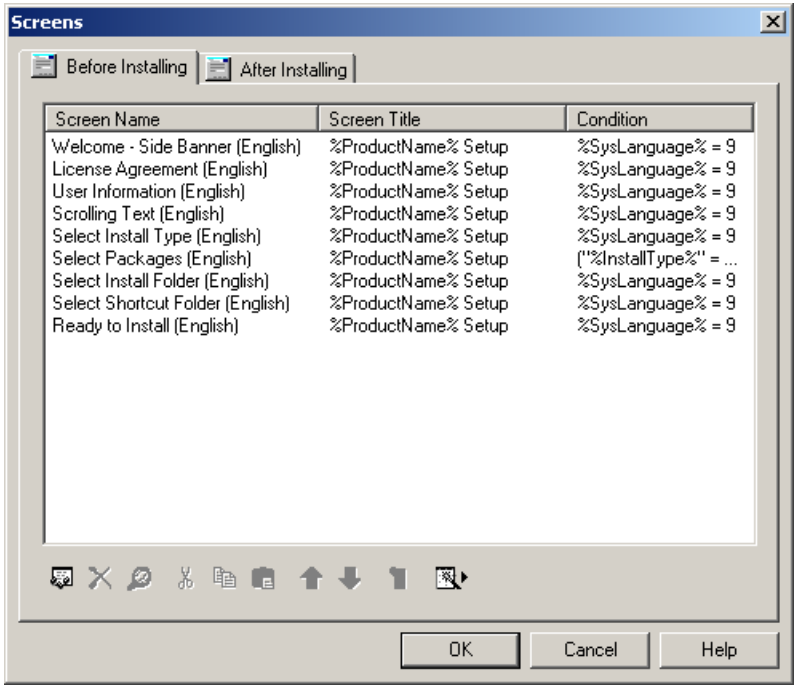
? For more information on assigning files to packages, see page 196.

Once you've assigned your files to the packages, you need to give your users a way to select those packages at run time. This is done by adding a *Select Install Type* screen and a *Select Packages* screen to your project.

Because you selected the **Support install types** option in the Project Wizard, both of these screens were already automatically added to your project. All you need to do is configure those screens to use the packages you added.

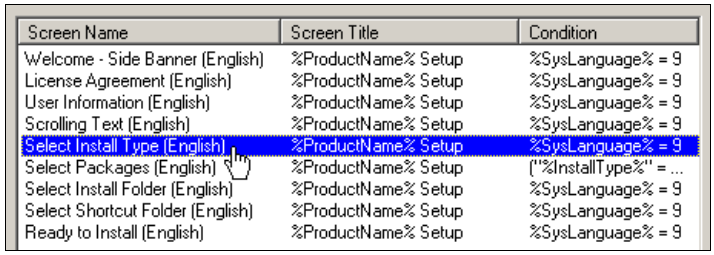
To configure the *Select Install Type* screen:

1. Select **Design | Screens** from the menu. This will open the *Screens* dialog.



The Screens dialog

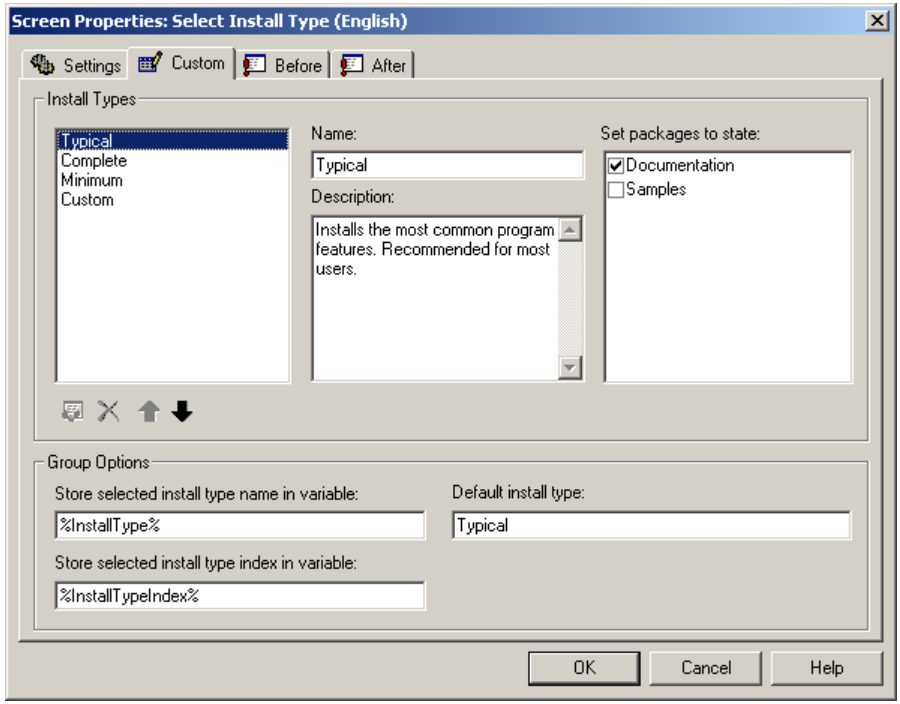
2. Select the "Select Install Type (English)" line. This line represents the *Select Install Type* screen on the *Screens* dialog.



Selecting the Select Install Type screen

3. Press the **Properties** button ( ). This will open the *Screen Properties* dialog for the *Select Install Type* screen.

4. Select the Custom tab. This is where you can configure the custom settings for the *Select Install Type* screen that you selected.



The Custom tab of the Screen Properties dialog for the Select Install Type screen

5. Select the "Typical" install type in the list on the left side of the Custom tab.

   Note that the name and description for the install type you select in the list are displayed in the **Name** and **Description** fields to the right.

6. Use the **Set packages to state** list on the right side of the Custom tab to select the packages that you want enabled when the user selects this install type.

   Any packages in the list that have check marks beside them will be enabled when the user selects the "Typical" install type at run time. Since the Typical install type is used to select the most often-used features, select the packages that you feel most users will want to install.

7. Select the "Complete" install type in the list on the left.

   When users select this install type, they want to install everything, so enable all of the packages in the **Select packages to state** list.

8. Select the "Minimum" install type in the list on the left.

   When users select this package, they only want the bare essentials installed, so enable only the essential packages in the **Select packages to state** list.

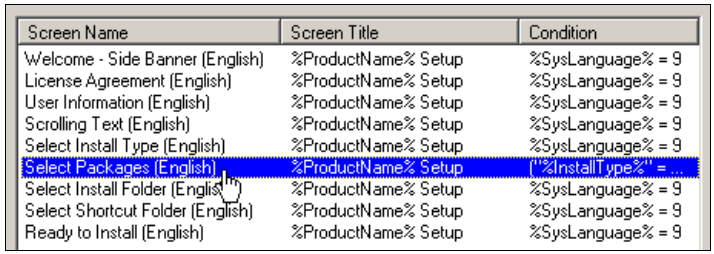
9. Select the "Custom" install type in the list on the left.

   When users select this package, they will be allowed to choose which individual packages they want to install. (This happens on the *Select Packages* screen, which we'll get to in a moment.) Use the **Select packages to state** list to select which packages will be enabled on the *Select Packages* screen by default.

10. Press **OK** to close the *Screen Properties* dialog and return to the *Screens* dialog.

Next you need to configure the *Select Packages* screen.
To configure the *Select Packages* screen:

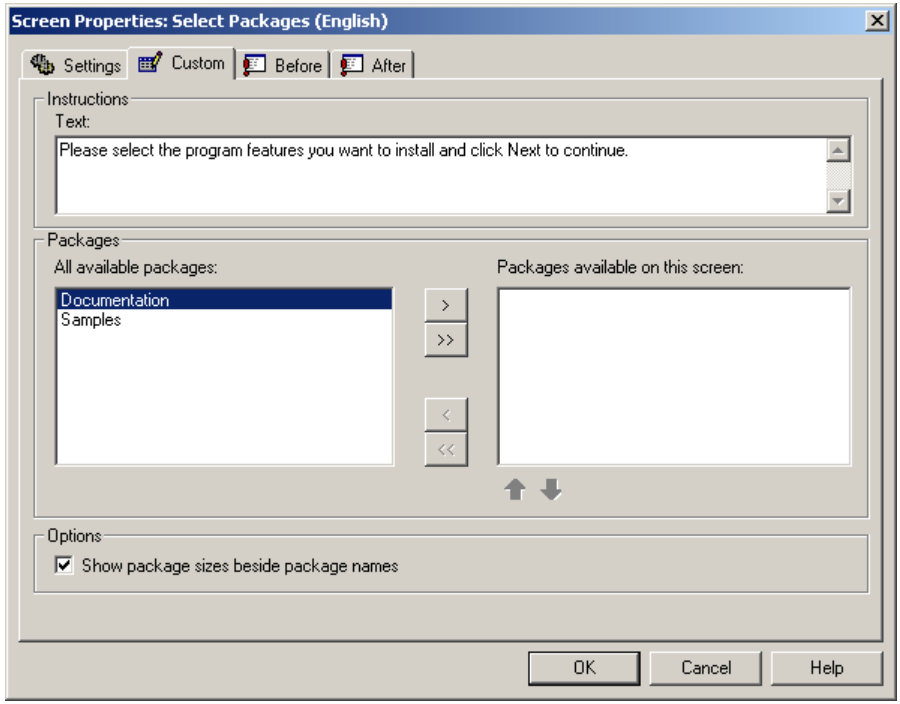
1. Select the "Select Packages (English)" line on the *Screens* dialog. This line represents the *Select Packages* screen.



Selecting the Select Packages screen

2. Press the **Properties** button ( ). This will open the *Screen Properties* dialog for the *Select Packages* screen.

3. Select the Custom tab. This is where you can configure the custom settings for the *Select Packages* screen that you selected.



The Custom tab of the Screen Properties dialog for the Select Packages screen

4. Press the **Move all** button (). This will move all of the packages from the **All available packages** list on the left to the **Packages available on this screen** list on the right.

5. Press **OK** to close the *Screen Properties* dialog and return to the *Screens* dialog.

6. Press **OK** to close the *Screens* dialog.

The *Select Install Type* screen and the *Select Packages* screen are now ready for the user to use at run time.

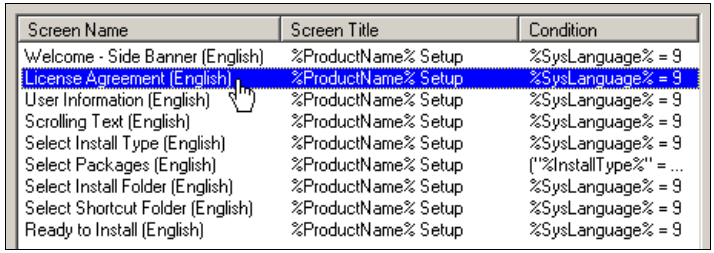
Step 6: Customize the Screens

You might want to edit the other screens in your installer, too. A good candidate is the *License Agreement* screen, which starts out with a default license agreement that you need to replace with your own legalese.

To replace the text on the *License Agreement* screen:

1. Select **Design | Screens** from the menu. This will open the *Screens* dialog.

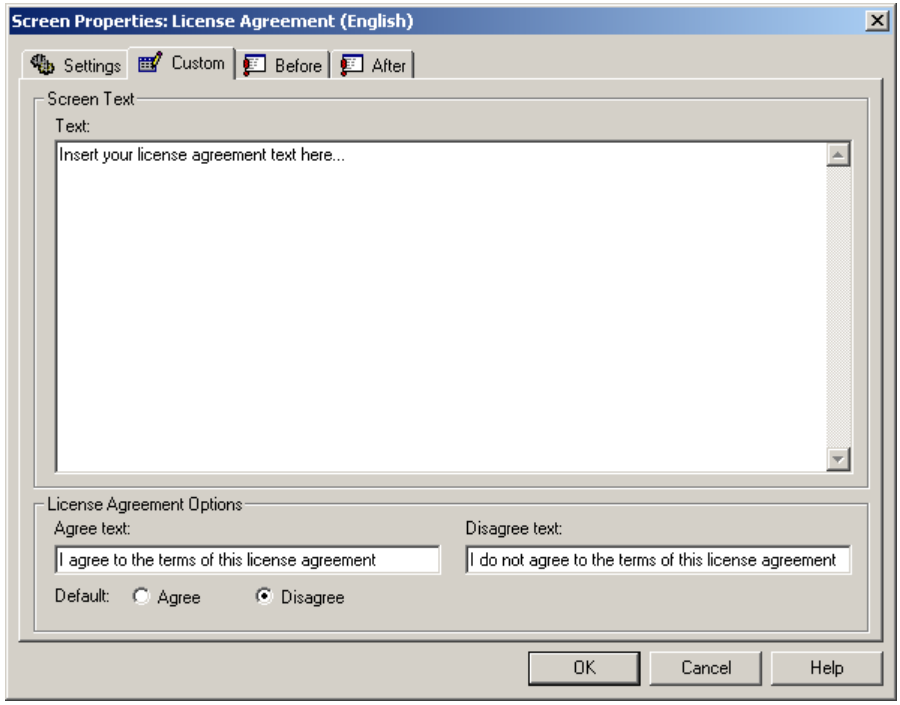
2. Select the *License Agreement* screen.



Selecting the License Agreement screen

3. Press the **Properties** button (). This will open the *Screen Properties* dialog for the *License Agreement* screen.

4. Select the Custom tab. This is where you can configure the custom settings for the *License Agreement* screen that you selected.



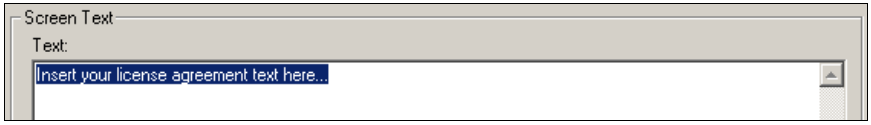
The Custom tab of the Screen Properties dialog for the License Agreement screen

5. Open your license agreement document in a document editor such as Microsoft Word or Notepad.

6. Copy your license agreement text to the clipboard (usually by selecting **Edit | Copy** or by pressing the Ctrl+C hotkey).

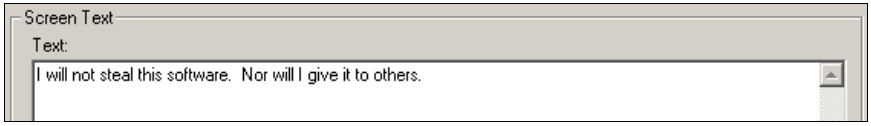
7. Go back to the *Screen Properties* dialog for the *License Agreement* screen in Setup Factory and click on the **Text** field.

8. Use the Ctrl+A hotkey to select all of the default license agreement text. This will highlight the text in the Text field so it can be replaced with your new license agreement text in the next step.



Out with the old...

9. Press the Ctrl+V hotkey to paste your license agreement into the **Text** field. This will replace the default license agreement text that you highlighted in the previous step.



...and in with the new

10. Press **OK** to close the *Screen Properties* dialog.

11. Press **OK** to close the *Screens* dialog.

Step 7: Add Any Required Actions

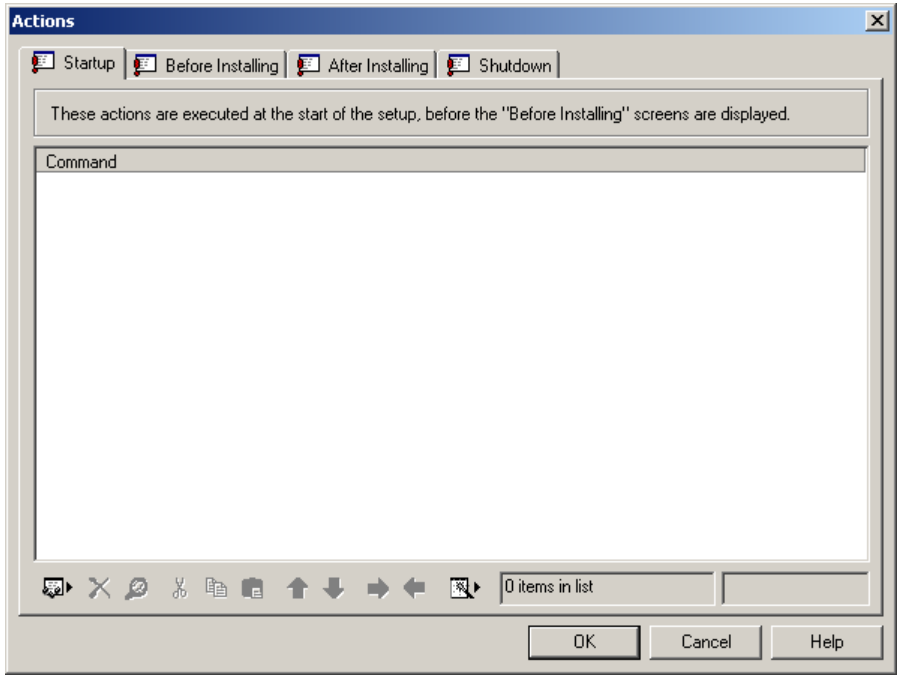
Setup Factory features over 50 powerful and flexible actions that you can use to *really* customize your installer. One of the actions that you will probably use often is the Modify Registry action.

In this step, we'll show you how to store the installation path in the Registry so that future installers can retrieve this value and use it to install files to the same location.

To add this Modify Registry action:

1. Select **Design | Actions** from the menu. This will open the *Actions* dialog, which is one of the places where you can add actions to your project.

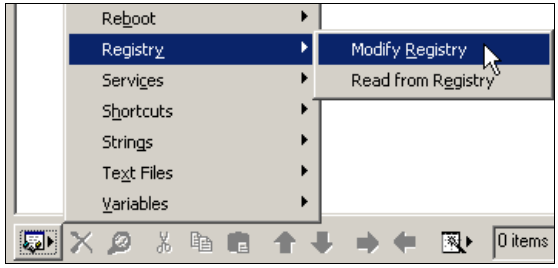
   Each tab on the *Actions* dialog corresponds to a different time during the installation process when actions can be performed.



The Actions dialog

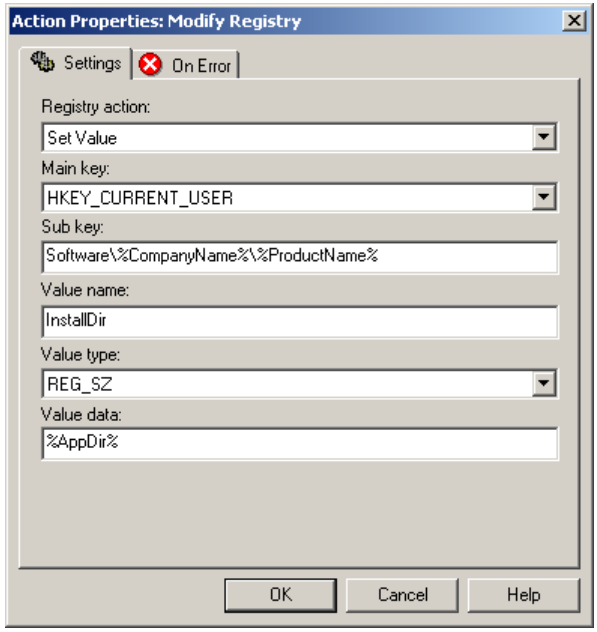
2. Switch to the After Installing tab. This way, the action will be performed after all the files in your project are installed.

3. Press the **Add Action** button ( ). This will pop up a list of action categories right next to the **Add Action** button.



The Modify Registry item in the Registry category

4. Navigate to the Registry category and select the Modify Registry action. This will open the *Action Properties* dialog for the Modify Registry action.



The Action Properties dialog for the Modify Registry action

5. Select "Set Value" in the **Registry action** field.

6. Select "HKEY_CURRENT_USER" in the **Main key** field.

7. Enter "Software\%CompanyName%\%ProductName%" in the **Sub key** field.

8. Enter "InstallDir" in the **Value name** field.

9. Select "REG_SZ" in the **Value type** field.

10. Enter "%AppDir%" in the **Value data** field.

11. Press **OK** to add this Modify Registry action.

SEE ALSO

? For more information on actions, see page 159.

TIP

You can remove this installation path from the Registry when your software is uninstalled by adding a Modify Registry action to the After Uninstalling tab of the *Uninstall* dialog.

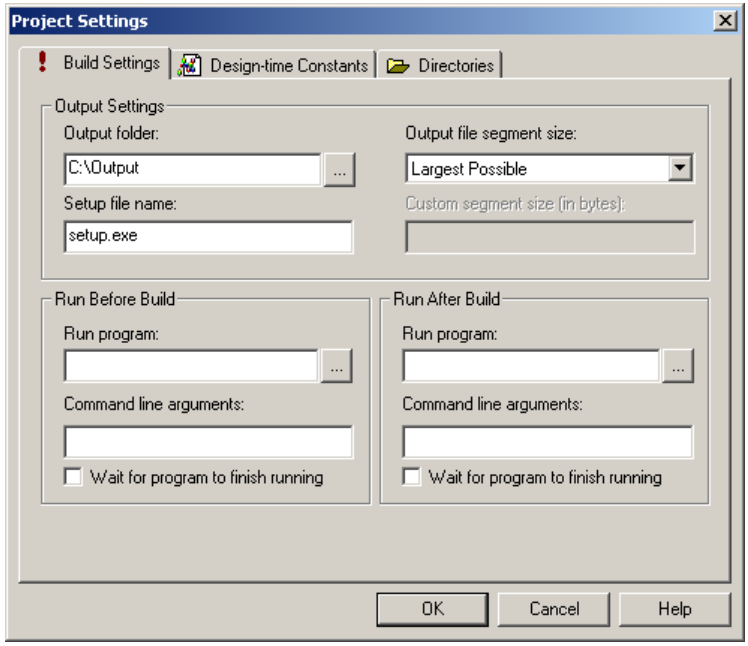
Step 8: Build the Setup Executable

Before building your setup executable, you might want to configure the build settings for your project. To do so:

1. Select **Project | Project Settings** from the menu. This will open the *Project Settings* dialog, where you can configure various options for your project.

2. On the Build Settings tab, enter the path to the **Output folder** where you want the setup executable to be built.

3. Enter the name you want your setup executable to have in the **Setup file name** field.



The Build Settings tab of the Project Settings dialog

4. Press **OK** to close the *Project Settings* dialog.

SEE ALSO

? For more information on these project settings, see page 100.

Now it's time to see your work in action!
To build the setup executable:

1. Select **Project | Build** from the menu.

2. When asked if you want to start the build process, select **Yes**.

If all goes well, the setup executable will be generated in the output folder, ready for you to test and distribute.

SEE ALSO

? For more information on testing and distributing your installer, see page 254.

Step 9: All Done!

Congratulations! You have just created an installer.

In many cases, this is all the functionality of Setup Factory that you will ever need. But if you love to tinker, don't worry—there's plenty more that you can do with Setup Factory.

Chapter 6

Working with Projects

What Are Project Files?

Setup Factory uses project files to store your work between sessions. These project files (`.sf6` files) contain the same information that goes into the setup executable when it's built.

Think of the project files as saving the work you do in Setup Factory at design time. When you load a project file into Setup Factory, you're returning Setup Factory to the same state it was in when that project file was saved.

NOTE

Setup Factory project files end with a `.sf6` file extension.

Starting a New Project

To start a new project, select **File | New** from the menu or press the Ctrl+N hotkey. The new project replaces the current project in Setup Factory. If you've made any changes to the current project, you will be asked if you'd like to save the changes before continuing.

Opening an Existing Project

To open an existing project, select **File | Open** from the menu or press the Ctrl+O hotkey. This project will replace the current project in Setup Factory. If you've made any changes to the current project, you will be asked if you'd like to save the changes before continuing.

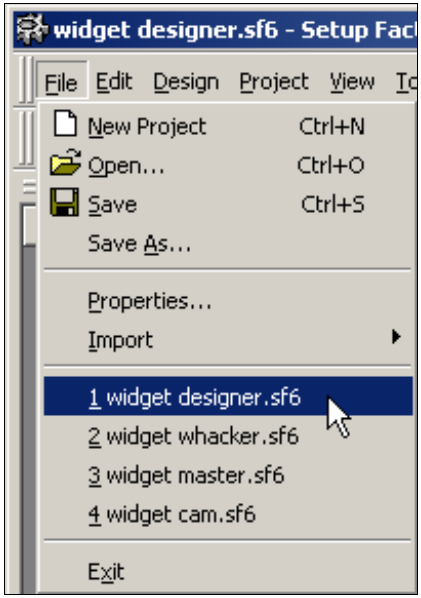
Saving the Current Project

To save the current project, select **File | Save** from the menu or press the Ctrl+S hotkey.

To save the project with a different filename, select **File | Save As**.

Reopening a Recent Project

You can access the four most recently saved project files directly from the **File** menu.



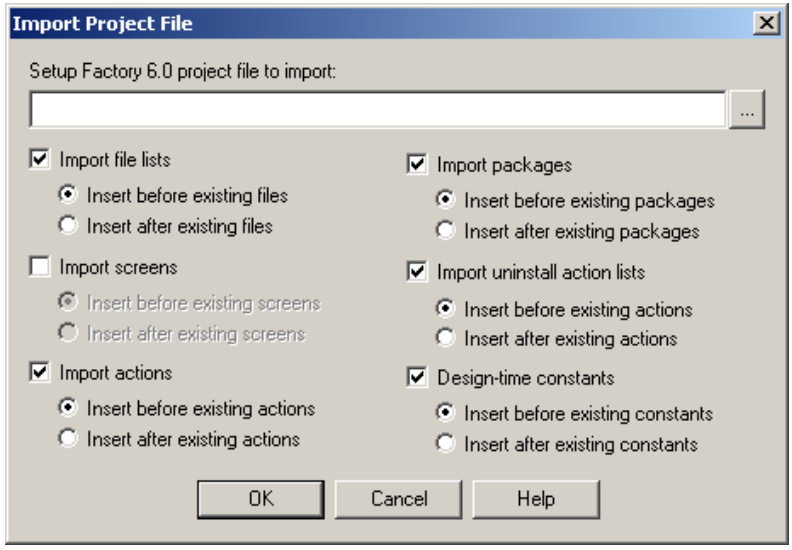
The list of recent project files in the File menu

TIP

The name of the current project file is displayed in Setup Factory's title bar.

Importing a Project

This feature allows you to merge two or more projects together. You can import a project file into the current project by selecting **File | Import** from the menu to open the *Import Project File* dialog.

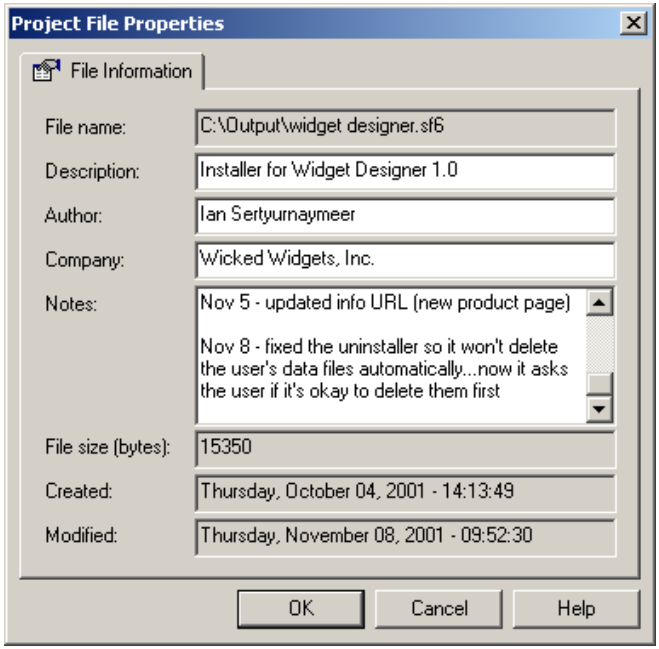


The Import Project File dialog

You can use the *Import Project File* dialog to control which parts of a project file you want to import into the current project. You can also control whether the imported parts are inserted before the corresponding parts of the current project, or appended after them.

Viewing and Editing Project File Properties

You can view and edit the project file properties by selecting **File | Properties** from the menu to open the *Project File Properties* dialog. The project file properties are for your information only—your users will never see the information that you enter here.



The Project File Properties dialog

Using the Project Wizard

You can use the Project Wizard to walk you through the steps required to build a simple Setup Factory project.

To do so, simply select **Project | Project Wizard** from the menu, and follow the instructions. You can also start the Project Wizard by pressing the **Project Wizard** button () on the Project toolbar.

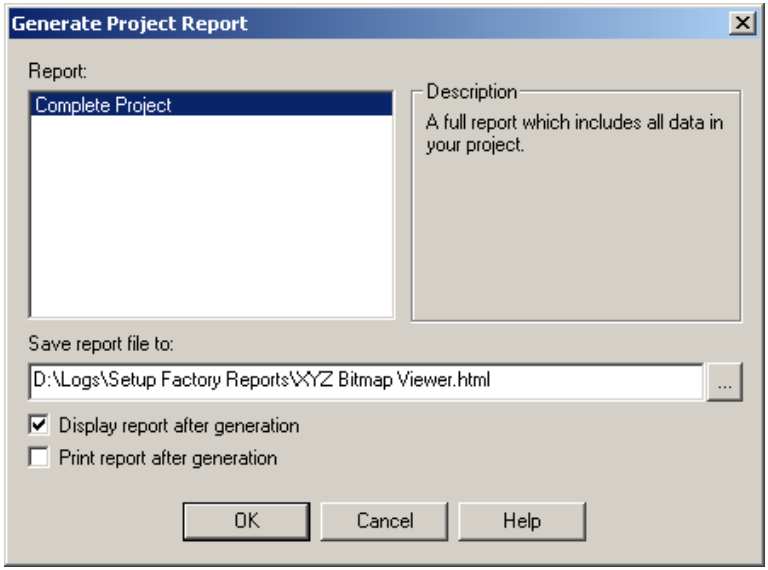
Generating a Project Report

A project report is an HTML file containing various information about a project, from information about the project file itself, to details on all of your design-time settings. The project reports are built from template files, which are located in the `Reports` sub-folder of your Setup Factory program directory.

You can customize the existing report types and even create your own report templates for Setup Factory to use. When a report is generated, specific keywords in the template files are replaced with information about the current project. Aside from the keywords, these templates are standard HTML files, so you can easily edit them with your favorite text editor. Project reports can contain as much or as little information on the current project as you like.

To generate a report on the current project:

1. Select **Project | Generate Report** from the menu. This will open the *Generate Project Report* dialog.



The Generate Project Report dialog

2. Select the type of report that you want to generate in the **Report** list. A description of the selected report type will appear to the right of the list.

3. Enter the full path and filename of the report file that you want to generate in the **Save report file to** field, or press the **Browse** button (...) to select a filename using the *Save As* dialog.

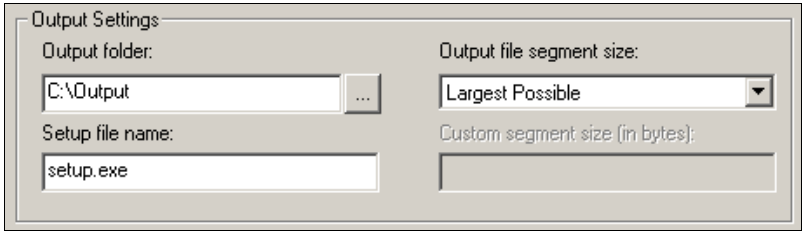
4. *Optional:*
   If you want to automatically view the report file in your default browser, select the **Display report after generation** check box.

5. *Optional:*
   If you want to automatically print the file with your default printer (using your default HTML browser), select the **Print report after generation** check box.

6. Press **OK** when you're ready to generate the report.

Project Build Settings

You can configure the build settings for the current project on the Build Settings tab of the *Project Settings* dialog. To access the *Project Settings* dialog, select **Project | Settings** from the menu.



The Output Settings section on the Build Settings tab

Changing the Output Folder

You can control where the setup executable will be generated by editing the path in the **Output folder** field. Alternatively, you can press the **Browse** button (...) to select a folder using the *Select Folder* dialog.

TIP

You can set the default output folder for new projects in the design environment preferences (see page 62).

Changing the Setup Executable Filename

You can change the name that will be given to the setup executable by editing the value in the **Setup file name** field.

Changing the Output File Segment Size

Setup Factory can output the setup executable as a single contiguous file, or it can break it up into several parts or "segments" of a specific size. This feature allows you to "span" a very large setup executable across two CD-ROMs, or to break an installer into pieces small enough to email without being rejected due to mail server size limits.

To output your installer into multiple segments, simply set the **Output file segment size** field to the maximum size you want each segment to have. You can use one of the default segment sizes, or you can select the "Custom" option and provide your own segment size in the **Custom segment size (in bytes)** field.

TIP

To span an executable across multiple floppy disks, leave the **Output file segment size** field set to "Largest Possible" and point the **Output folder** directly to your floppy drive. During the build process, Setup Factory will automatically prompt you to insert a new floppy as each disk is filled to capacity.

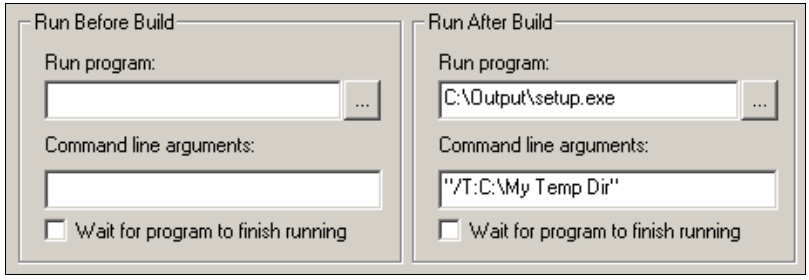
TIP

You can modify the existing segment size options or add your own segment sizes to the list by editing the `segmentsizes.ini` file in the `Data` sub-folder of your Setup Factory program directory.

Automatically Running a Program Before or After the Build Process

Setup Factory can automatically execute an external program before and after the installer is built. For example, you could update a readme file with the current build date using a search-and-replace utility before Setup Factory builds the installer. Or, you could automatically call a batch file to copy the setup executable to a different directory right after it's built.

Or you might want to automatically run your installer to test it. To do so, you would enter the output path and filename for the setup executable in the **Run program** field in the **Run After Build** section on the Build Settings tab of the *Project Settings* dialog.



The Output Settings section on the Build Settings tab

To automatically run a program before or after the build process:

1. Enter the full path to the program you want to run in the appropriate **Run program** field—i.e., the one in the **Run Before Build** section or the one in the **Run After Build** section—on the Build Settings tab of the *Project Settings* dialog.

2. *Optional:*
   Enter any command line arguments you want to pass to the program in the **Command line arguments** field.

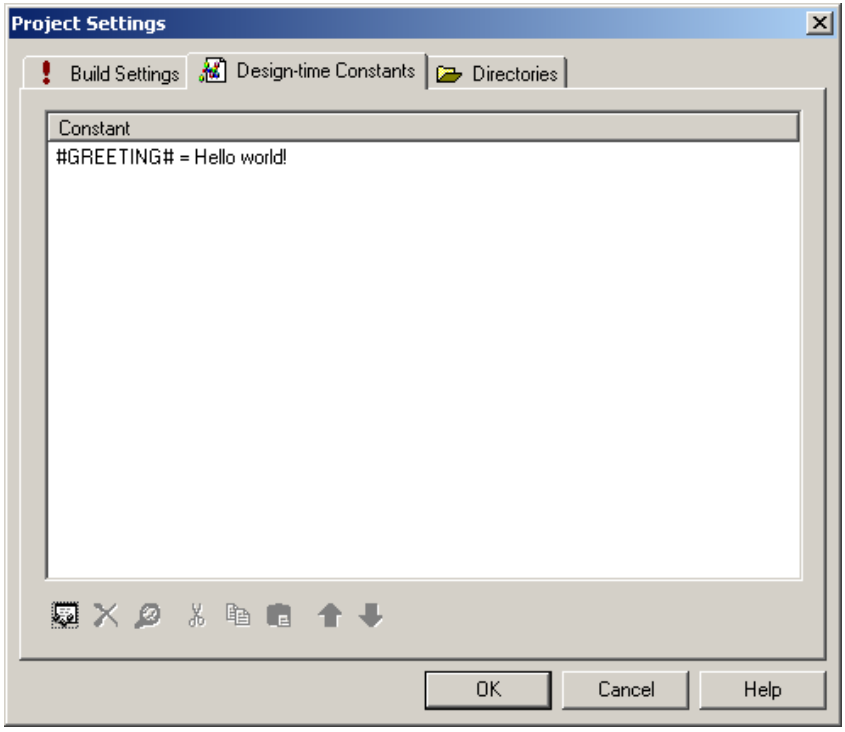
3. *Optional:*
   If you want Setup Factory to wait for the program to finish before continuing with the build process, select the **Wait for program to finish running** check box.

Design-time Constants

You can add, remove and edit design-time constants for the current project on the Design-time Constants tab of the *Project Settings* dialog. To access the *Project Settings* dialog, select **Project | Settings** from the menu.

SEE ALSO

? For more information on design-time constants, see page 37.

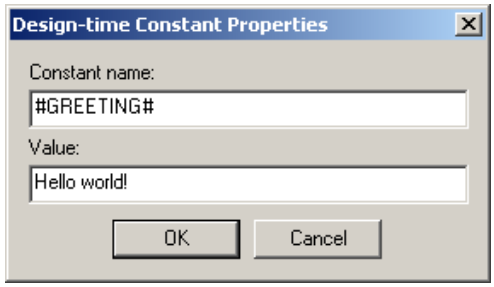


The Design-time Constants tab of the Project Settings dialog

Adding Design-time Constants

To add a design-time constant:

1. Press the **Add** button ( ) or use the Insert hotkey. This will open the *Design-time Constant Properties* dialog.



The Design-time Constant Properties dialog

2. Enter a name for the constant in the **Constant name** field. We strongly recommend that you begin and end all constant names with a number (#) sign, like so: #MY_CONSTANT#.

3. Enter a value in the **Value** field. This value will replace the constant name at build time.

Removing Design-time Constants

To remove a design-time constant:

1. Select the design-time constant that you want to remove on the Design-time Constants tab of the *Project Settings* dialog.

2. Press the **Remove** button ( ) or use the Delete hotkey.

Editing Design-time Constants

To edit a design-time constant:

1. Select the design-time constant that you want to edit.

2. Press the **Properties** button () or use the CTRL+P hotkey to open the *Design-time Constant Properties* dialog.

TIP

You can also double-click on a constant on the Design-time Constants tab to open the *Design-time Constant Properties* dialog.

3. Edit the **Constant name** and **Value** fields as desired.

Base Directories

When files are added to a Setup Factory project, Setup Factory tries to set a "best guess" value for the destination folder of each file. It does this by comparing the file's path to the base directory for the tab the files were added to.

The base directory is the path to a folder on *your* system that represents the application directory (%AppDir%) on the *user's* system. There is one base directory for the Archive tab, and another base directory for the CD-ROM tab. When you add files to your project, Setup Factory uses the appropriate base directory setting to determine where the files should be installed at run time.

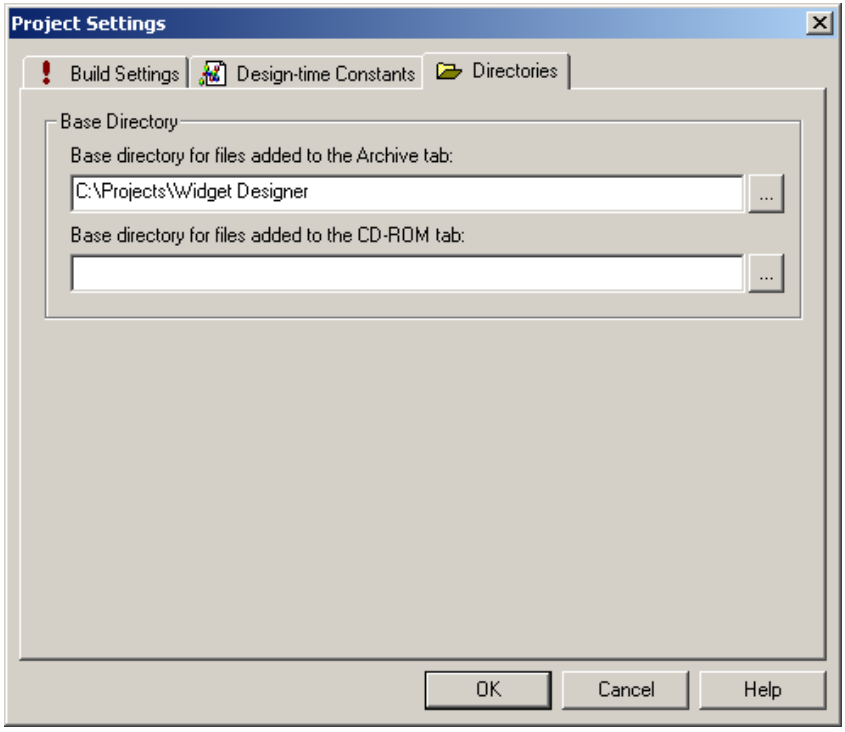
When you add a file, any part of the file's source path that matches the base directory is essentially replaced by %AppDir% in the destination path for the file.

For example, if you add `C:\Products\Foo\Data\settings.ini` to your project, and the base directory is `C:\Products\Foo`, the file's destination path will automatically be set to `%AppDir%\Data\settings.ini`.

SEE ALSO

? For more information on the base directory, see page 109.

You can view or edit the current base directory for the Archive tab and the CD-ROM tab on the Directories tab of the *Project Settings* dialog. To access the *Project Settings* dialog, select **Project | Settings** from the menu.



The Directories tab of the Project Settings dialog

NOTE

The base directory only matters when files are added. Changing the value in either field has no effect on files that have already been added to your project.

Changing the Base Directory for the Archive Tab

To change the current base directory for the Archive tab, edit the path in the **Base directory for files added to the Archive tab** field, or press the **Browse** button (...) to select a folder using the *Select Folder* dialog.

Changing the Base Directory for the CD-ROM Tab

To change the current base directory for the CD-ROM tab, edit the path in the **Base directory for files added to the CD-ROM tab** field, or press the **Browse** button (...) to select a folder using the *Select Folder* dialog.

NOTE

The corresponding base directory setting is reset (deleted) whenever you remove all of the files from the Archive tab or CD-ROM tab.

Building the Current Project

To build the current project, select **Project | Build** from the menu. (You can also press the **Build** button ( ) on the Project toolbar, or use the F7 hotkey.)

You will be asked to confirm that you want to start the build process. If you select **Yes**, the *Status* dialog will appear and the build process will proceed. If all goes well, the setup executable will be generated in the output folder, ready for you to test and distribute.

SEE ALSO

? For more information on testing and distributing your installer, see page 254.

TIP

You can start the build process from a batch file by running Setup Factory with the Unattended Build (/B) command line option. (See page 260.)

Chapter 7

Working with Files

Installing the right files to the right places is the key to performing a successful installation. The files that make up your software need to get from your development system to the appropriate locations on the user's computer. Regardless of whether you're distributing files on CD-ROMs, floppies, or in a single, downloadable setup executable, Setup Factory allows you to install all of your files with the utmost precision.

NOTE

Setup Factory will install files exactly where you tell it to. In order to design a successful installer, you need to know where your files need to go.

The Project Window

The project window is where all of the files that you add to your Setup Factory project are listed. You can use the project window to highlight specific files and view or edit their properties.

There are two tabs on the project window: the Archive tab, and the CD-ROM tab.

The Archive Tab

The Archive tab is for files that you want compressed and stored in the setup executable. In other words, this tab is for files that will be included *in* your installer.

You should add files to the Archive tab when you want them physically included in the setup executable file.

The CD-ROM Tab

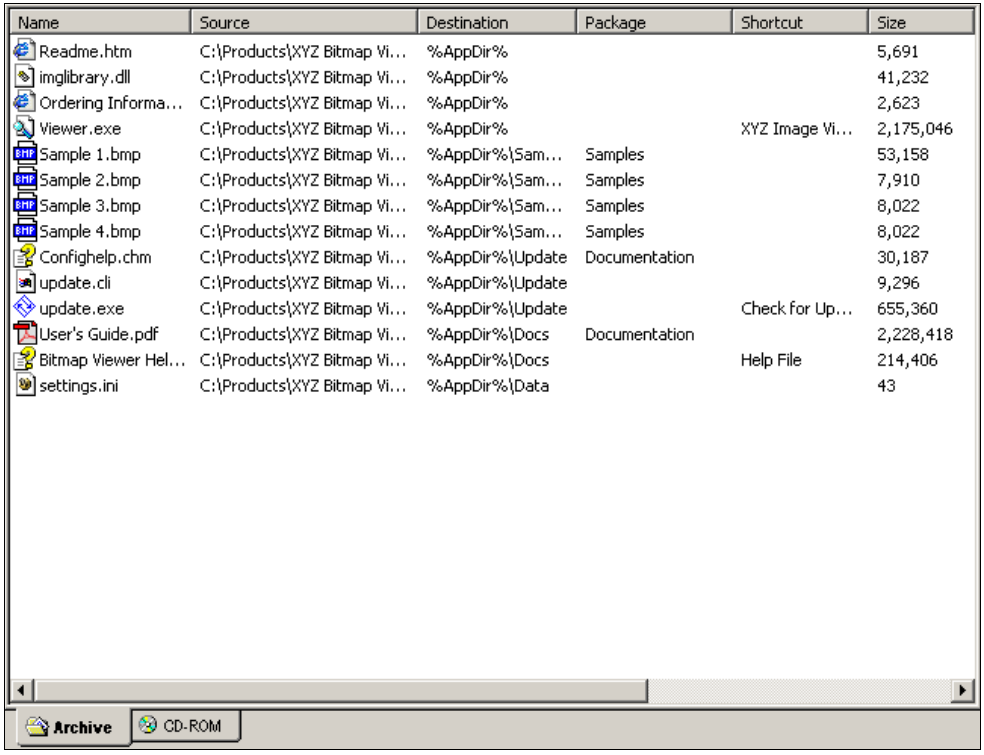
The CD-ROM tab is for files that you *don't* want compressed and stored in the setup executable. In other words, this tab is for files that will be distributed *with* your installer.

You should add files to the CD-ROM tab when you *don't* want them physically included in the setup executable file.

When you add files to the CD-ROM tab, you're essentially telling Setup Factory: "don't worry about how these files got there, just look for them *here* and install them to *there*."

NOTE

You can use the CD-ROM tab even if you aren't distributing your software on a CD-ROM. Setup Factory doesn't care how the files on the CD-ROM tab end up on the user's system; it just needs to be able to find them where you said they would be at run time.



A list of files on the Archive tab of the Setup Factory project window

Before Adding Files

Before you start adding files to your project, make sure you have all of your files properly installed on your system. Your local directory structure affects the destination paths that Setup Factory assigns to files as they are added to your project, so it's important to have your application's directory structure laid out on your system the same way you want it to end up on the user's.

SEE ALSO

? For more information on preparing your directory structure, see page 47.

Adding Files

There are two ways to add files to your Setup Factory project: you can add files from within Setup Factory, or you can drag and drop files onto the project window.

IMPORTANT

! When you add files to your project, they are not actually moved or copied from their original locations—instead, Setup Factory references or "links to" the files at their original locations.

Adding Files From Within Setup Factory

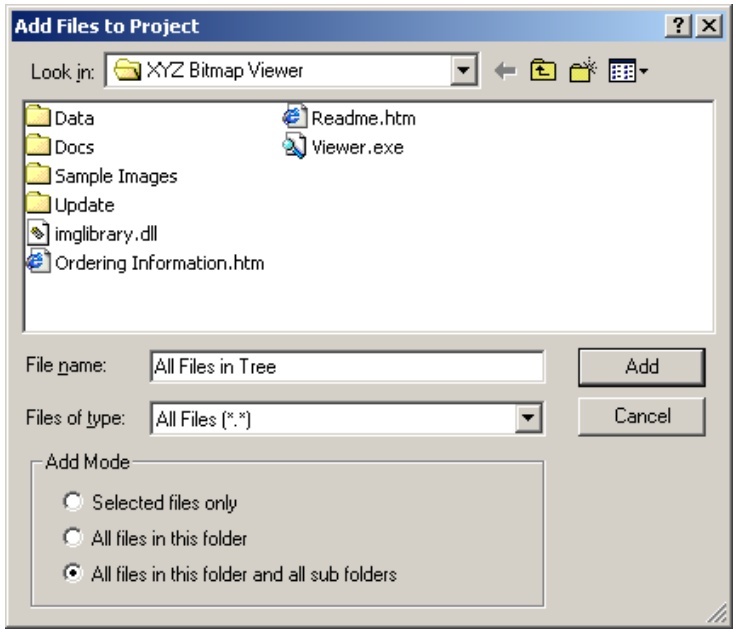
To add files from within Setup Factory:

1. Select the Archive tab on the project window if you want the files to be included in the setup executable, or select the CD-ROM tab on the project window if the files will be distributed separately.

2. Select **Edit | Add Files** from the menu.

   (You can also press the **Add Files** button (), use the Insert hotkey, or right-click on the project window and select **Add Files** from the context menu.)

   This will open the *Add Files to Project* dialog.



The Add Files to Project dialog

3. Select the add mode that you want to use:

   The **Selected files only** option will only add the files that you select with the *Add Files to Project* dialog.

   The **All files in this folder** option will add all of the files in the folder that the *Add Files to Project* dialog is currently displaying.

   The **All files in this folder and all sub-folders** option will add all of the files in the current folder, and all of the files in any sub-folders as well.

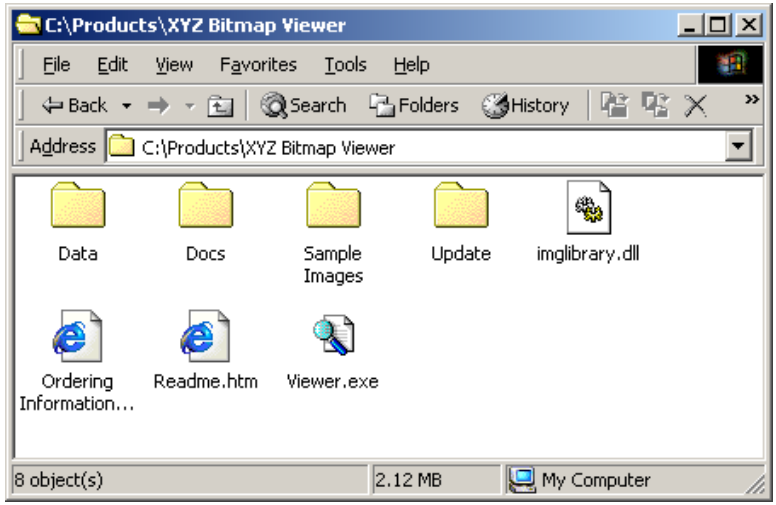
4. Select the files that you want to add, or navigate to the folder where all the files (and possibly sub-folders) that you want to add are located.

5. Press the **Add** button to add the files to your project.

Dragging Files Onto the Project Window

To add files by dragging and dropping them:

1. Open the folder where your files are located.



The "C:\Products\XYZ Bitmap Viewer" folder in Windows

2. Select the files that you want to add to your project.

TIP

You can drag and drop folders, too. When you select a folder, all of the files in that folder and in all of its sub-folders are selected as well.

3. With your files still selected, hold the left mouse button down and drag the files over the Setup Factory project window.

4. Let go of the mouse button to "drop" the files onto the project window.

How the Base Directory is Converted to %AppDir%

When files are added to a Setup Factory project, Setup Factory tries to set a "best guess" value for the destination folder of each file. It does this by comparing the file's path to the *base directory* for the tab the files were added to.

The base directory is the path on your system that corresponds to the path where your application will be installed. In other words, it's the local representative of %AppDir%. (%AppDir% is the built-in variable that gets set to the installation path when the user selects an install folder at run time.)

Setup Factory automatically picks a base directory for you the first time you add any files to your project. If you add a single file, Setup Factory chooses the folder that the file is located in as the base directory. If you add multiple files at the same time, Setup Factory chooses the first folder that all of the files have in common as the base directory.

Once a base directory is set, whenever a file is added to a Setup Factory project, any part of the source path that matches the base directory is replaced with "%AppDir%".

For instance, if the current project's base directory for the Archive tab was `C:\Projects\Widget Designer` and you added the following files to the Archive tab:

```
C:\Projects\Widget Designer\Widget Designer.exe
C:\Projects\Widget Designer\Data\Parts.dat
C:\Projects\Widget Designer\Docs\Widget Designer.html
```

...the destination paths for the files would be set to:

```
%AppDir%\Widget Designer.exe
%AppDir%\Data\Parts.dat
%AppDir%\Docs\Widget Designer.html
```

If you add a file that isn't located somewhere beneath the base directory—so the path to the base directory isn't found in the path to the source file—Setup Factory sets the file's destination path to just "%AppDir%".

NOTE

The base directory is reset (i.e. cleared) whenever you remove all the files from a tab. In other words, the base directory for the Archive tab is reset when you remove all the files from the Archive tab, and the base directory for the CD-ROM tab is reset when you remove all the files from the CD-ROM tab. This means that if you remove all the files from one of the tabs, and then proceed to add files to it again, Setup Factory will pick a new base directory based on the files you just added.

TIP

You can also set the base directory manually. This can be helpful if you have two different directory structures that you want to install to the same place on the user's system.

For example, you might have two different versions of your software installed in two different directory structures on your system (a full version and a demo version, perhaps). If you wanted to install them both to the same place on the user's system, you could add the files from the first directory structure, change the path of the base directory to point to the second directory structure, and then add the files from the second directory structure to your project. (Once the files were added, you could assign them to packages to control which version is installed at run time.)

To set the base directory manually, select **Project | Settings** from the menu, and edit the appropriate path on the Directories tab of the *Project Settings* dialog.

SEE ALSO

? For more information on how to set the base directories manually, see page 105.

Removing Files

To remove files from your Setup Factory project:

1. Select the files that you want to remove on the project window.

2. Select **Edit | Remove Files** from the menu.

   (You can also press the **Remove Files** button (), use the Delete hotkey, or right-click on the project window and select **Remove Files** from the context menu.)

File Properties

You can use the *File Properties* dialog to view and edit the settings for any file in your project.

To access the *File Properties* dialog:

1. Select a file on the project window.

2. Select **Edit | File Properties** from the menu.

   (You can also press the **File Properties** button (), use the CTRL+Enter hotkey, or right-click on the file and select **File Properties** from the context menu.)

   This will open the *File Properties* dialog, where you can view and edit the properties of the file you selected.

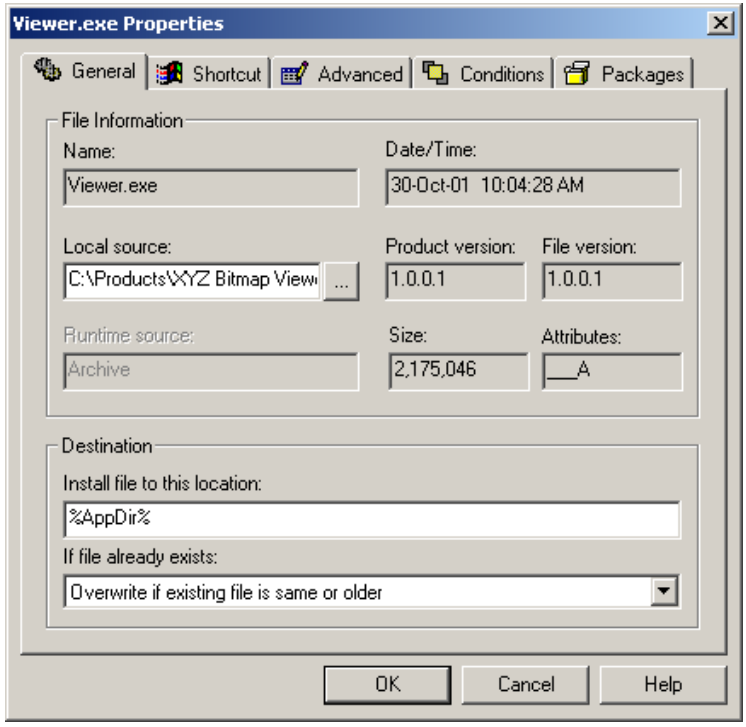
TIP

You can also access the *File Properties* dialog by double-clicking on a file in the project window.

There are five tabs on the *File Properties* dialog: General, Shortcut, Advanced, Conditions and Packages.

The General Tab

The General tab is where you will find information about the file itself, such as its name, date/time stamp, size, and other common attributes. This is also where you can view or edit the source path, the destination path, and the overwrite settings for the file.



The General tab of the File Properties dialog

There are two possible source paths for every file: the local source path, and the runtime source path.

Local source path

The local source path is the path to the file on your development system at design time. In other words, this is where Setup Factory expects to find the file on your system.

Runtime source path

The runtime source path is the path to the file on the user's system at run time. In other words, this is where Setup Factory expects to find the file at the beginning of the installation process.

NOTE

The runtime source path only applies to files on the CD-ROM tab. The files on the Archive tab are always located in the setup executable archive at run time.

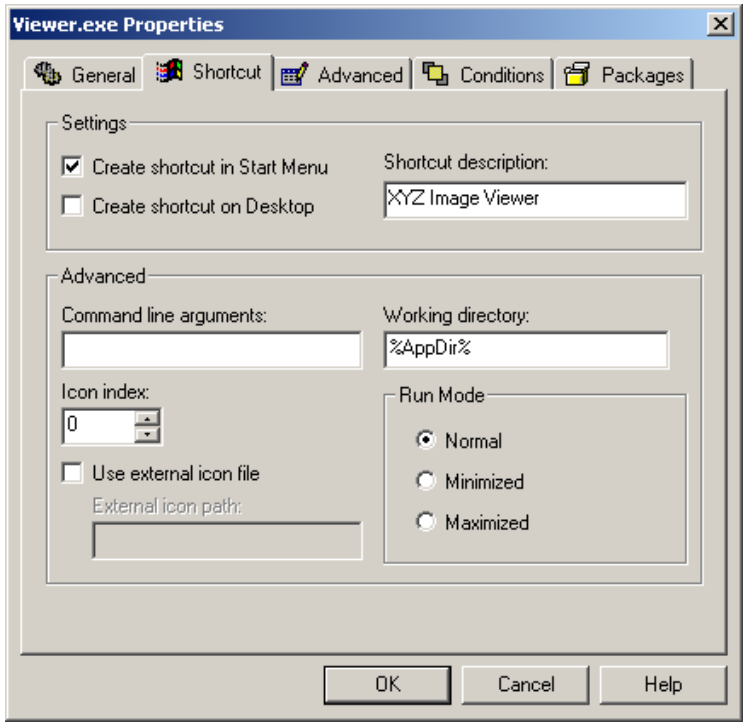
The destination path is the path that the file will be installed to on the user's system. You can change this path by editing the value in the **Install file to this location** field.

SEE ALSO

? For more information on the General tab, please consult the Command Reference.

The Shortcut Tab

The Shortcut tab is where you can configure the shortcut options for the selected file. You can use this tab to have Setup Factory automatically create a shortcut icon for the file in the user's Start menu or on the user's desktop at run time.



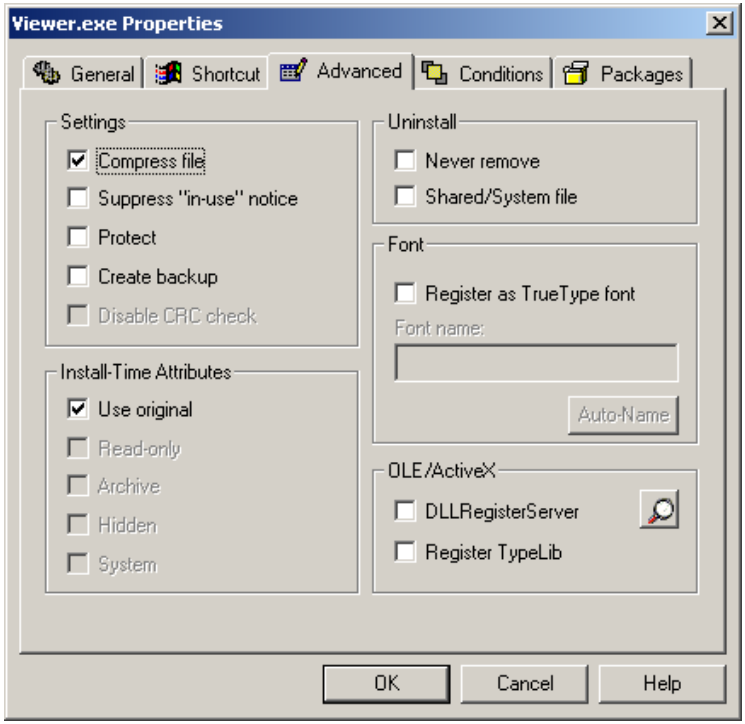
The Shortcut tab of the File Properties dialog

SEE ALSO

? For more information on the Shortcut tab, please consult the Command Reference.

The Advanced Tab

The Advanced tab is where you can configure advanced options for the selected file. This is where you can override certain default installation settings, such as whether the file will be compressed before storing it in the setup executable, whether it will be uninstalled automatically by your installer's uninstall routine, and what attributes will be assigned to the file after it's installed. You can also specify whether the file should be registered as a TrueType font, or even whether it should be registered as a shared OLE or ActiveX component using DLLRegisterServer.



The Advanced tab of the File Properties dialog

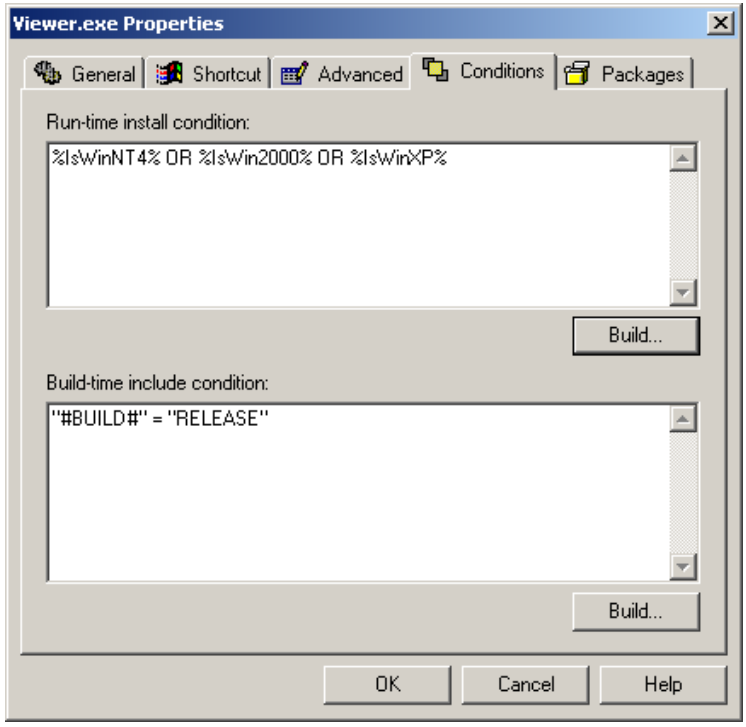
SEE ALSO

? For more information on the Advanced tab, please consult the Command Reference.

The Conditions Tab

The Conditions tab is where you can edit the run-time and build-time conditions for the selected file.

The run-time condition is an expression that determines whether the selected file is installed at run time. The build-time condition is an expression that determines whether the file is included in the setup executable at run time.



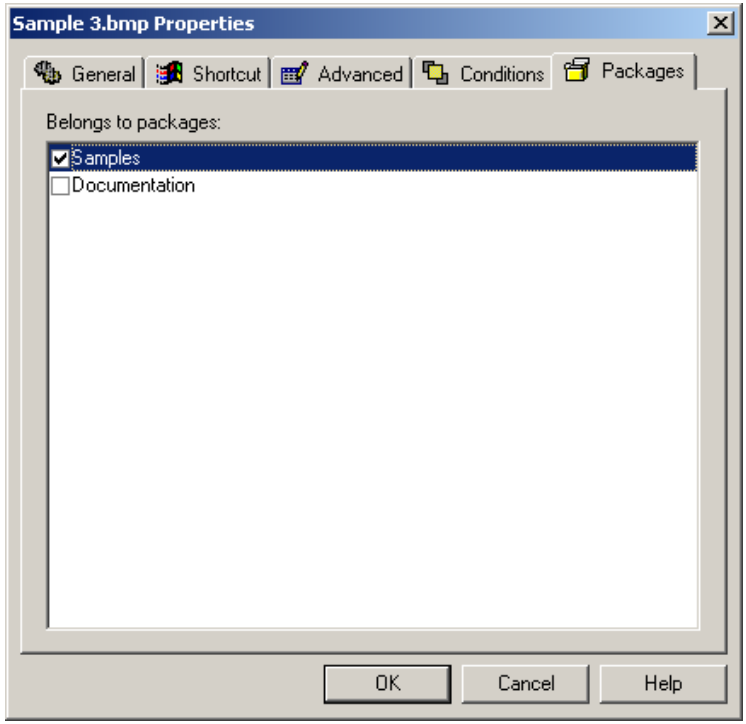
The Conditions tab of the File Properties dialog

SEE ALSO

? For more information on run-time and build-time conditions, see pages 40-41 and 221-223. For more information on the Conditions tab, please consult the Command Reference.

The Packages Tab

The Packages tab is where you can select the packages that you want the currently selected file to belong to. The Packages tab lists all of the packages that have been configured in your project. You can add the currently selected file to one or more packages by clicking on the check boxes in this list.



The Packages tab of the File Properties dialog

The file will be assigned to the packages that have check marks beside them. Clicking on an empty check box adds a check mark, and clicking on a box that already has a check mark clears it.

An empty check box looks like this: ☐ Samples

A selected check box looks like this: ☑ Samples

Multiple File Properties

You can use the *Multiple File Properties* dialog to edit the properties of more than one file in your project at a time.

To access the *Multiple File Properties* dialog:

1. Select more than one file on the project window. (You can select multiple files by pressing and holding the Ctrl key or the Shift key while you click on the files.)

2. Select **Edit | File Properties** from the menu.

   (You can also press the **File Properties** button (), use the CTRL+Enter hotkey, or right-click on the file and select **File Properties** from the context menu.)

   This will open the *Multiple File Properties* dialog, where you can view and edit the properties of the files you selected.

The *Multiple File Properties* dialog is similar to the *File Properties* dialog, but there are some important differences:

- The *Multiple File Properties* has four tabs instead of five; the General tab and the Shortcut tab are combined into a single General/Shortcut tab.

- Entering text in a field changes the corresponding setting in all of the selected files.

- Check boxes on the *Multiple File Properties* dialog have three states: enabled (), disabled (), and mixed (). The mixed state preserves the settings for that check box in all of the selected files.

- The Conditions tab has two special drop-down lists that let you choose how to apply any text you enter in the **Run-time install condition** and **Build-time include condition** fields to the existing conditions. You can replace the existing conditions with the new text, prepend the text to each of the conditions with either an AND or an OR, append the text to the end of each condition with an AND or an OR, or leave the existing conditions unchanged.

Missing Files

Since Setup Factory only records an informational link to each file, and doesn't actually maintain a copy of the file itself, it's possible for files to go "missing" from Setup Factory's point of view. For example, if a file in your project is moved to another directory, Setup Factory will no longer be able to find it at its original location. The same thing happens when a file is renamed or deleted. Setup Factory only knows to look for a file at the place where it was when you "showed" the file to Setup Factory by adding it to your project.

Although Setup Factory might not know where a missing file has ended up, it definitely knows when a file is missing. Whenever a file in your project can't be found at its original location, Setup Factory displays the file's information on the project window in red instead of black. The red color makes it easy for you to see which files in your project are missing.

If you find that your files are suddenly playing hide-and-seek with Setup Factory, try to remember if you've made any changes to your files recently. If you moved the files, you can try moving them back. If you renamed the files, you can restore the original names. If you deleted the files, you'll have to replace them.

If Setup Factory still shows the files in red after you've corrected the situation, you probably just need to refresh the display. To do so, simply select **View | Refresh** from the menu, or use the F5 hotkey. Refreshing the display causes Setup Factory to re-examine the location of every file in your project.

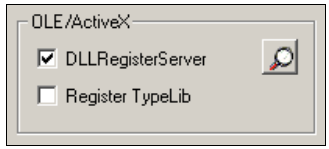
NOTE

Setup Factory automatically refreshes the display for you when you open a project or initiate the build process.

If you moved, renamed, or deleted the files on purpose, and you want Setup Factory to use the files at their new locations, or remember the files by their new names, or just forget about the past and move on, you'll need to either remove the files from your project and add them back in again, or change the local source path on the *File Properties* dialog for each file. The only way to "show" Setup Factory where a new file is on your development system is to add the file to your project.

Registering Files

There are two ways that you can register files (such as ActiveX controls) in Setup Factory. The easiest way is to simply select the **DLLRegisterServer** check box on the Advanced tab of the *File Properties* dialog for the file you want to register.



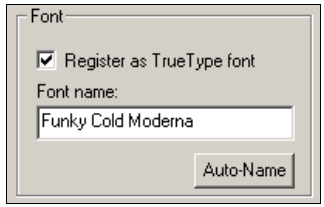
The alternative method is to use a Register File action to register a file after it has been installed.

SEE ALSO

? For more information on the **DLLRegisterServer** option and the Register File action, please consult the Command Reference.

Registering Fonts

There are two ways that you can register TrueType fonts in Setup Factory. The easiest way is to select the **Register as TrueType font** check box on the Advanced tab of the *File Properties* dialog for the file you want to register. The file must be a valid Windows TrueType font (`.ttf`) file.



The alternative method is to use a Register Font action to register a `.ttf` file after it has been installed.

SEE ALSO

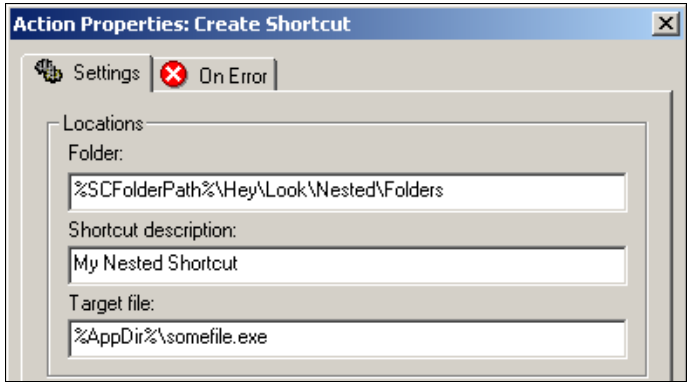
? For more information on the **Register as TrueType font** option and the Register Font action, please consult the Command Reference.

Nested Shortcuts

A nested shortcut is simply a shortcut file that is located in a shortcut sub-folder—i.e., a shortcut file in a folder that is located or "nested" in another folder.

You can easily create nested shortcuts with Setup Factory 6.0 by using a Create Shortcut action. Simply include the nested folders in the path that you provide in the **Folder** field on the *Action Properties* dialog for the Create Shortcut action.

In other words, just enter the full path to where you want the shortcut file to end up, and if any of the folders in the path don't exist, Setup Factory will automatically create them for you.



Create Shortcut action settings to create a nested shortcut

SEE ALSO

? For more information on the Create Shortcut action, please consult the Command Reference.

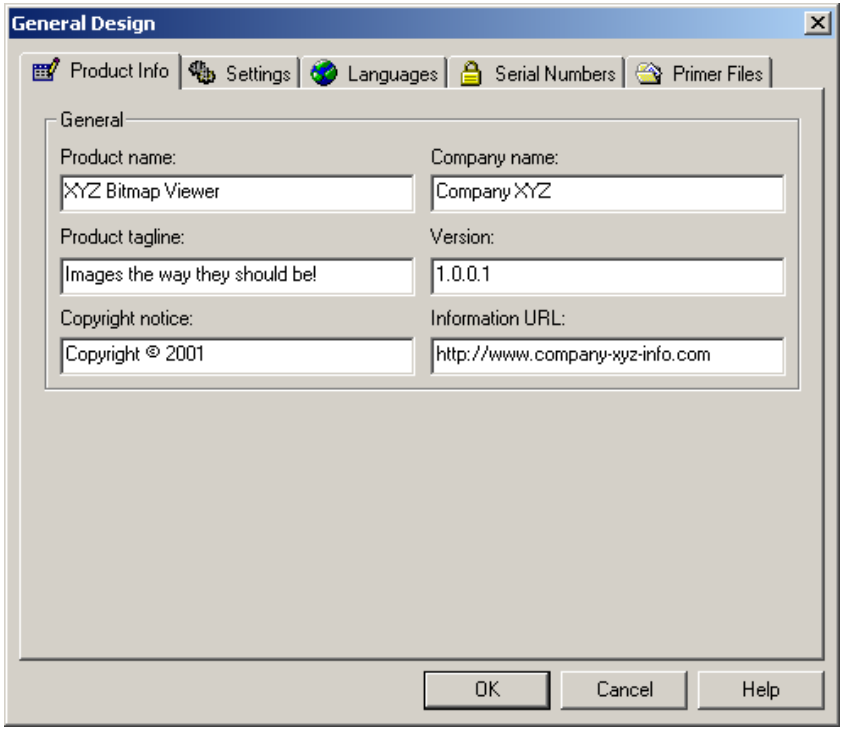
Chapter 8

General Design

The *General Design* dialog is where you can configure general design settings for your installer.

You can access the *General Design* dialog by selecting **Design | General Design** from the menu, or by clicking on the General Design icon on the shortcut bar.

There are five tabs on the *General Design* dialog: Product Info, Settings, Languages, Serial Numbers, and Primer Files.

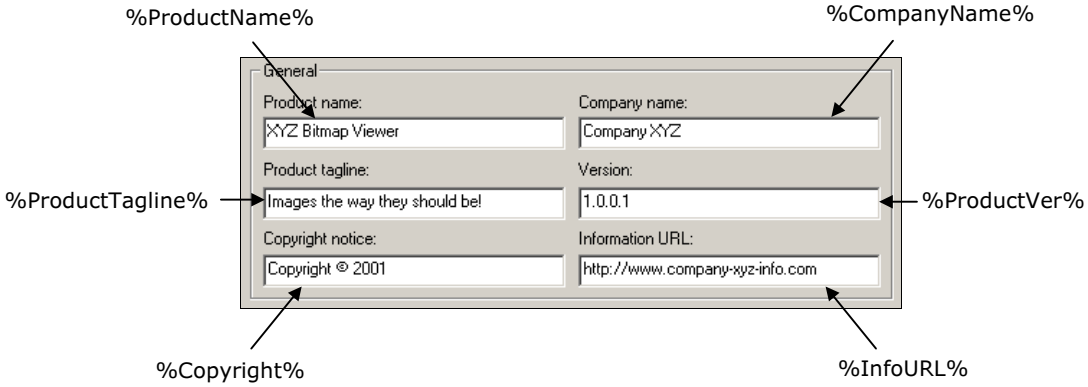


The Product Info tab of the General Design dialog

The Product Info Tab

The Product Info tab is where you can provide general information about your product. This includes the name of your product, your company name, a product tagline or "marketing slogan," a version string, your copyright notice, and the web site address where more information about your product can be found.

The information you provide on the Product Info tab is made available throughout your project in the form of six built-in variables: %ProductName%, %CompanyName%, %ProductTagline%, %ProductVer%, %Copyright%, and %InfoURL%.



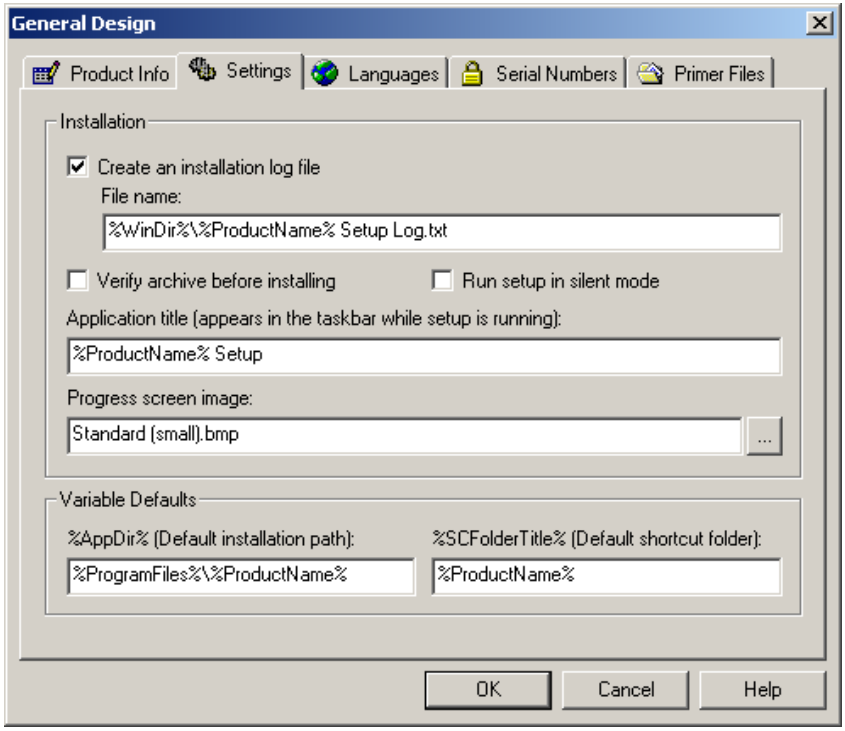
SEE ALSO

? For more information on variables, see pages 35 and 211.

For more information on the Product Info tab, please consult the Command Reference.

The Settings Tab

The Settings tab is where you can configure general installation options for your installer. This includes whether a log of the installation process will be generated, whether the setup will run in silent mode, what the installer's name in the taskbar will be, and what the default values for the installation path (%AppDir%) and shortcut folder name (%SCFolderTitle%) will be.



The Settings tab of the General Design dialog

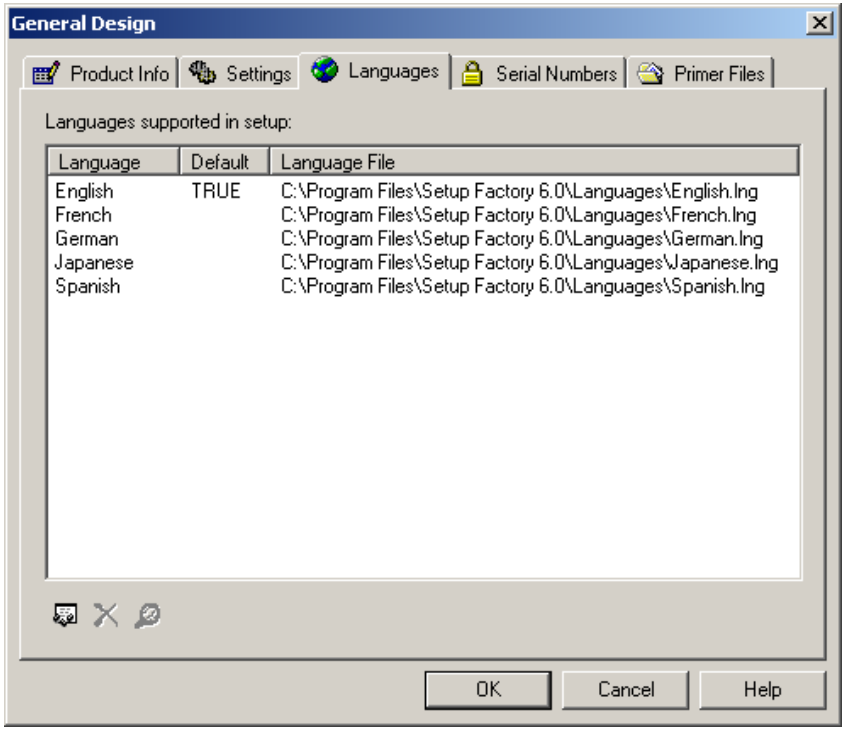
SEE ALSO

? For more information on the Settings tab, please consult the Command Reference.

The Languages tab

The Languages tab is where you can add language modules to your project and translate the various text messages used by your installer. You can use the Languages tab to edit the text for all the messages, buttons and prompts that can appear during an installation. These "built-in" messages are stored in Setup Factory language (`.lng`) files.

Language files are special INI files used by Setup Factory. Each language file corresponds to a specific Windows language ID, and includes all of the "built-in" messages for that language in plain text format. You can find the Setup Factory language files in your `C:\Program Files\Setup Factory 6.0\Languages` folder.



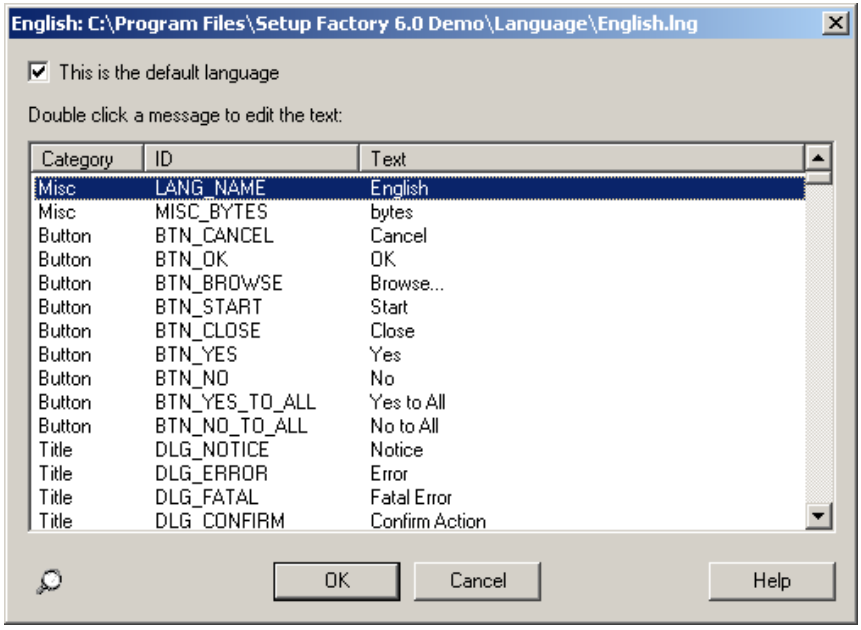
The Languages tab of the General Design dialog

Editing Messages

To edit the messages associated with a language:

1. Select the language you want to edit on the Languages tab.

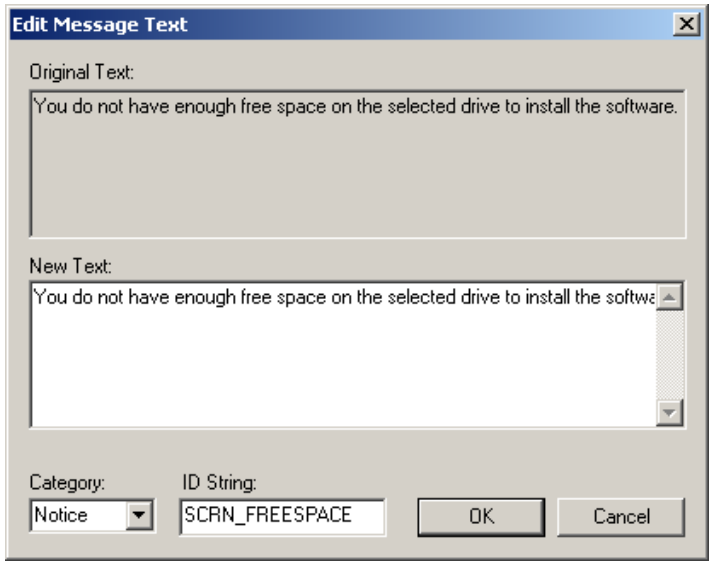
2. Press the **Properties** button ( ), or right-click on the language and select **Properties** from the context menu. This will open the *Language Module Messages* dialog.



The Language Module Messages dialog

3. Select the message you want to edit.

4. Press the **Properties** button ( ), or right-click on the message and select **Properties** from the context menu. This will open the *Edit Message Text* dialog.



The Edit Message Text dialog

5. Edit the message to your liking and press the **OK** button.

TIP

You can also edit messages using your text editor by editing a language file directly, and then adding the modified language file to your Setup Factory project.

You can create a new language module by copying one of the existing language files and then editing the copy.

Setting the Default Language

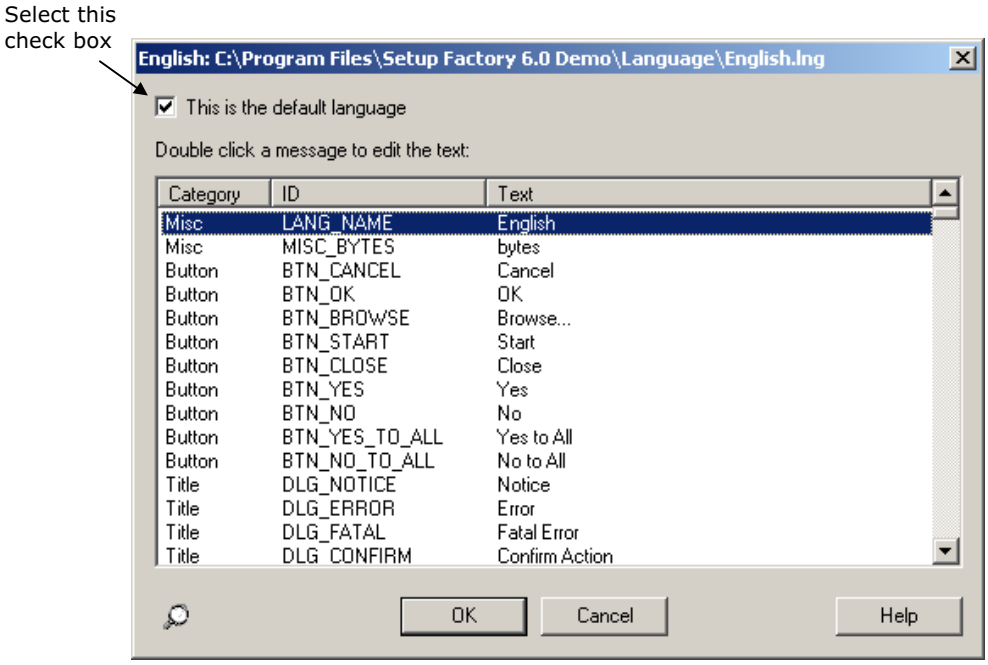
If the installer doesn't have a language module for the language being used on the user's system, the messages from the default language will be used instead.

To set the default language:

1. Select the language on the Languages tab that you want to use as the default language.

2. Press the **Properties** button ( ), or right-click on the language and select **Properties** from the context menu. This will open the *Language Module Messages* dialog.

3. Select the **This is the default language** check box to make this language the default language.



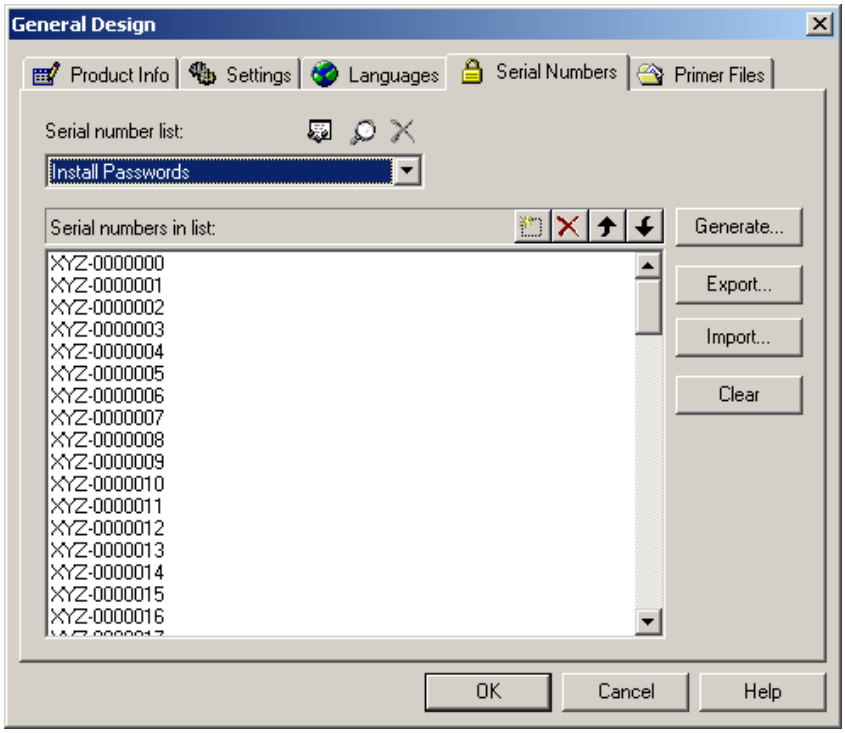
The Language Module Messages dialog

SEE ALSO

? For more information on the Languages tab, please consult the Command Reference.

The Serial Numbers Tab

The Serial Numbers tab is where you can create, delete and modify lists of serial numbers to include in your installer. These serial numbers can then be used with *Verify Serial Number* screens to control access to individual features or even the whole installation at run time.



The Serial Numbers tab of the General Design dialog

For example, you could use the Serial Numbers tab to generate a list of 20,000 serial numbers, and assign that list to a *Verify Serial Number* screen that is shown right after your *License Agreement* screen at runtime. Your users would be required to enter a valid serial number on that *Verify Serial Number* screen in order to proceed with the installation.

The Serial Numbers tab can be especially useful when you release new versions of your software. For instance, if you learned that a group of serial numbers had been leaked by a

warez site, you could remove those numbers from your lists in the next version of your installer. That way, users would no longer be able to use those stolen serial numbers to install your software.

IMPORTANT

! Although Setup Factory does its best to help secure the installation process, it can't keep your software safe by itself. The security of your installations will ultimately depend on how you issue and keep track of serial numbers.

Setup Factory can only secure the installation of your software; once your software is installed, it is beyond the protection of Setup Factory's security features.

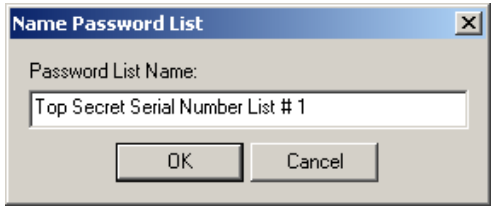
If security is a concern, make sure you take whatever other steps are necessary to protect your software.

Implementing serial numbers is a two step process. The first step is to create a list of serial numbers on the Serial Number tab.

Creating a List of Serial Numbers

To create a list of serial numbers:

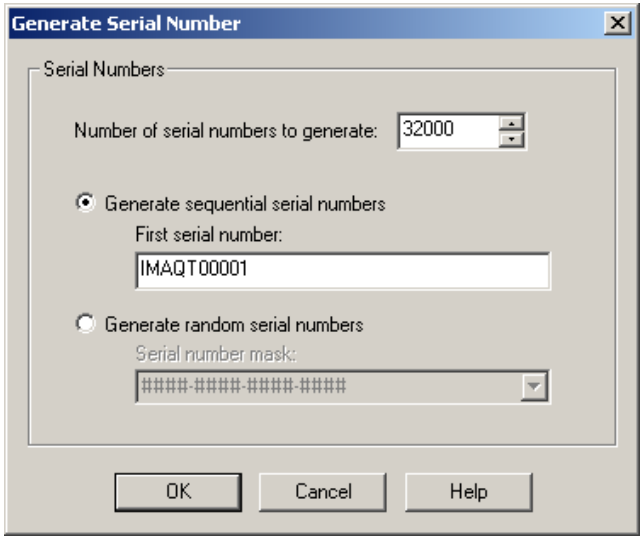
1. Press the **Add a new serial number list** button (). This will open the *Name Password List* dialog.



The Name Password List dialog

2. Enter a name for the new list and press **OK** to return to the Serial Numbers tab.

3. Press the **Generate** button (). This will open the *Generate Serial Number* dialog.



The Generate Serial Numbers dialog

4. Use the *Generate Serial Number* dialog to generate your serial numbers.

NOTE

You can have as many serial number lists in your project as you want.

Why more than one list?

Setup Factory allows you to create multiple serial number lists because there may be cases where you need more than one set of serial numbers to unlock different products or features. For example, you may want to distribute several related products in a single setup executable, while still requiring your users to purchase licenses for each product separately. By creating a separate list of serial numbers for each product, you can make the verification process product-specific. That way, the user couldn't just use the serial number from one product to install all of the others.

TIP

You can also import a list of serial numbers from a text file.

To do so, press the **Import** button on the Serial Numbers tab, and select the text file that you want to import the serial numbers from.

The text file must have one serial number per line, and each line can have a maximum length of 100 characters. For example, if your text file began with the following three lines, each line would become a separate serial number in the list:

```
139-EL33T-256-A
358-ID10T-444-H
358-IR11T-001-L
```

Note: you can also export a list of serial numbers to a text file by using the **Export** button.

Once you've created a list of serial numbers, you can add a *Verify Serial Number* screen to your installer and configure its settings on the Custom tab of the *Screen Properties* dialog to use the serial number list that you created.

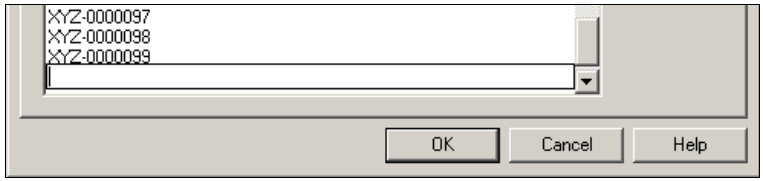
SEE ALSO

? For more information on how to add screens to your installer, see page 150. For more information on the Serial Numbers tab, please consult the Command Reference.

Adding a Serial Number to the List

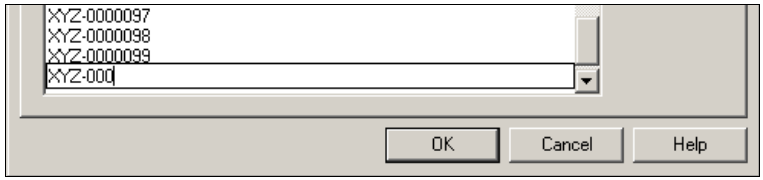
To add a new serial number to the list:

1. Press the **New** button ( ). This will insert a blank line at the end of the list, and position the cursor at the beginning of the line.



Ready to enter a new serial number

2. Type in the serial number.



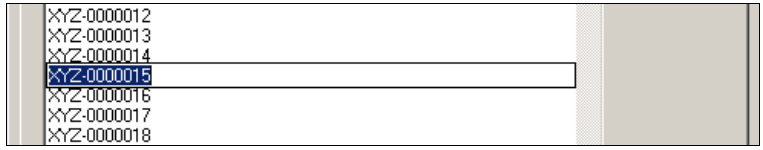
Typing in a new serial number

Changing a Serial Number in the List

To change an existing serial number in the list:

1. Double-click on the serial number you want to change.

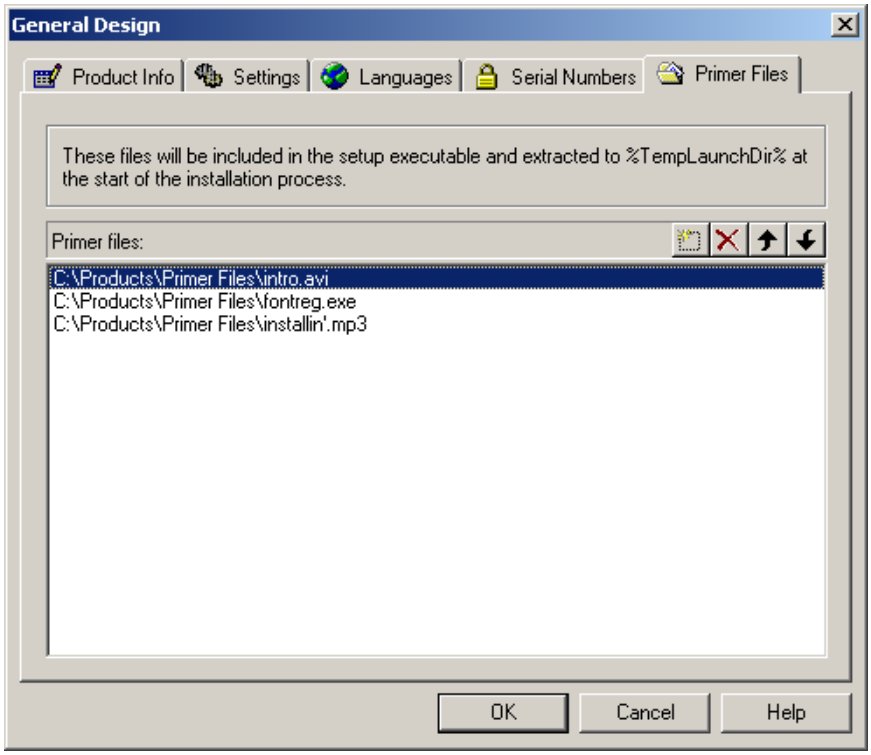
2. Use the Backspace, Delete, and Left or Right arrow keys to edit the serial number, or just type in a new serial number.



Editing a serial number

The Primer Files Tab

The Primer Files tab is where you can edit the list of primer files that will be included in the setup executable. The files in this list will be extracted from the setup executable *before* the installation process begins so they can be used at the very start of the installation process, right after the user runs the setup executable.



The Primer Files tab of the General Design dialog

SEE ALSO

? For more information on Primer Files, see page 43.

For more information on the Primer Files tab, please consult the Command Reference.

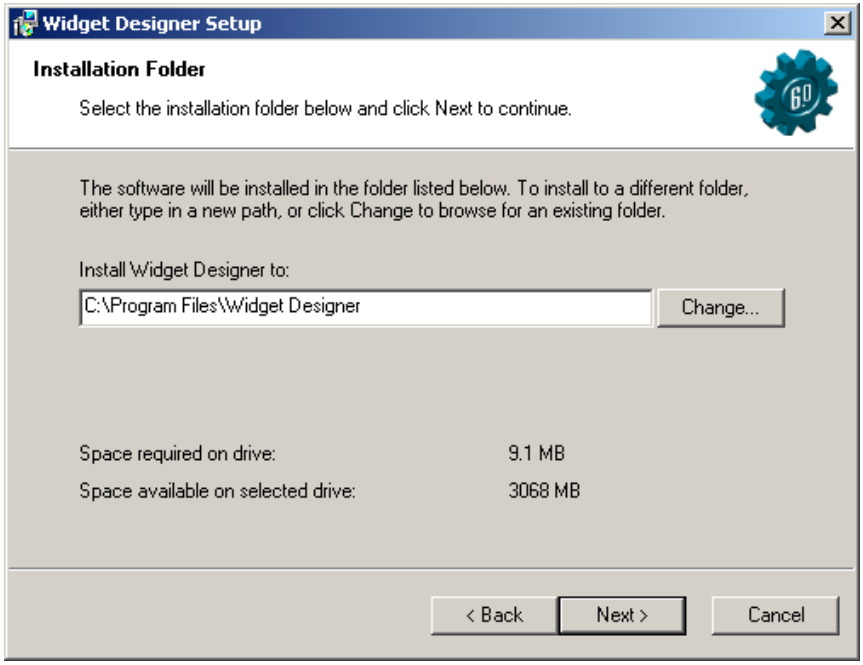
Chapter 9

Screens

Screens make up the visual user interface of your installer. They're what the user sees and interacts with throughout the installation process.

You can use screens to display information to the user, and you can use them to get information from the user as well.

The screens in Setup Factory use a familiar "wizard" style. Each screen either has an image on the left-hand side, with text and options on the right, or a banner along the top, with text and options below. The bottom of the screen has a combination of buttons like Next, Back, Finish, and Cancel, depending on the screen.



A Select Install Folder screen at run time

Screens in a Nutshell

Here's a basic overview of screens:

- Screens make up the visual interface of your installer.
- You can have as few or as many screens in your project as you want. In fact, you don't have to display any screens at all.
- Screens are displayed in the same order as they're listed on the *Screens* dialog.
- Screens on the Before Installing tab are displayed *before* files are installed.
- Screens on the After Installing tab are displayed *after* files are installed.
- Each screen has a *screen condition* that determines whether the screen will be shown.
- A screen is hidden when its screen condition evaluates to false.
- An optional "Help" button can be enabled on each screen.
- Actions can be performed immediately before or after any screen is displayed. They can also be performed when the user presses the Help button on a screen.
- You can add actions to the Before and After tabs of the *Screen Properties* dialog, and you can also add them to the *Help Button Actions* dialog.
- Information entered by the user on screens is usually stored in custom variables.

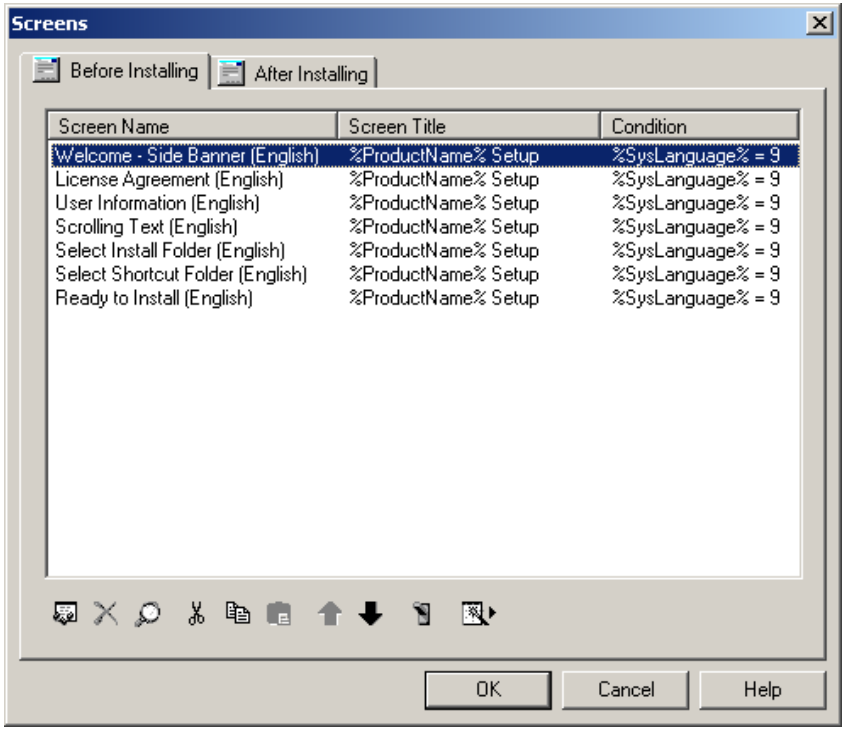
The Screens Dialog

The *Screens* dialog is where you can add, remove, arrange, preview, export and configure all of the screens in your Setup Factory project.

There are two tabs on the *Screens* dialog: the Before Installing tab, and the After Installing tab. These two tabs represent the two different times during the installation process when screens can be shown.

At run time, the screens are displayed in the same order as they're listed on these tabs.

You can access the *Screens* dialog by selecting **Design | Screens** from the menu, or by clicking on the Screens icon on the shortcut bar.



The Before Installing tab of the Screens dialog

The Before Installing Tab

The Before Installing tab lists the screens that will be displayed *before* the files in your project are installed. All of the screens on this tab will be shown before Setup Factory begins installing the files that you added to the Archive and CD-ROM tabs.

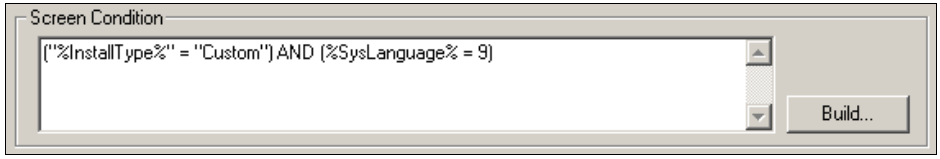
The After Installing Tab

The After Installing tab lists the screens that will be displayed *after* the files in your project are installed. All of the screens on this tab will be shown after Setup Factory is done installing the files that you added to the Archive and CD-ROM tabs.

Screen Conditions

A screen condition is simply an expression that determines whether a screen will be hidden or displayed at run time. If the expression is blank, or evaluates to a true result, the screen will be shown. If the expression evaluates to a false result, the screen will not be shown.

Each screen has its own screen condition that determines whether that screen will be shown. You can edit the screen condition on the Settings tab of the *Screen Properties* dialog.



The Screen Condition field at the bottom of the Settings tab

Screen Actions

Screen actions are simply actions that you add to the Before and After tabs on a *Screen Properties* dialog. Depending on which tab you add them to, these actions will be performed immediately before or after the associated screen is displayed at run time.

You can use these actions to make your installer respond even more intelligently to field situations. For example, you could use actions to process the information entered on one screen, and present the results to the user on the next. Or, you could use actions to compare the user's system information with what they entered on a *User Information* screen, and ask more questions if their information didn't match.

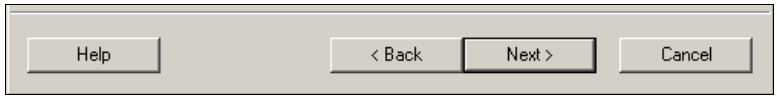
The full palette of Setup Factory actions is available on these action tabs—even advanced actions like Check Internet Connection and Download File HTTP.

NOTE

There is a Before and After tab for each screen in your project.

The Help Button

You can display an optional **Help** button on any screen in your installer. Each help button has an action list associated with it. When the user presses a **Help** button, the actions in that button's action list are performed.



The Help button at the bottom of a screen at run time

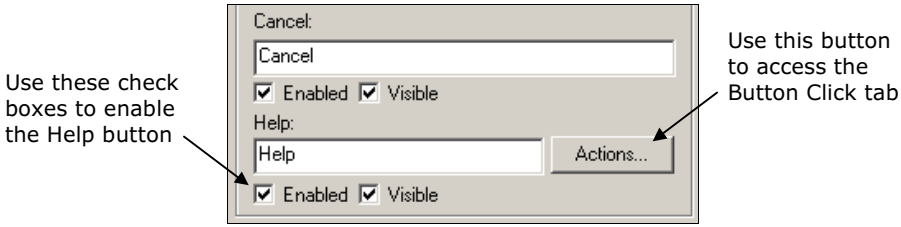
For example, you could use an Open Document action to open a Word document or HTML file with instructions on how to use the current screen—or even to play an audio file with verbal instructions for your users.

TIP

You can use the Help button to perform any Setup Factory actions you want. In fact, you can even change the name of the Help button and use it to serve a completely different function.

To enable the Help button on a screen, select the **Enabled** check box and the **Visible** check box for that button on the Settings tab of the *Screen Properties* dialog.

To access the *Help Button Actions* dialog, where you can edit the list of actions that will be associated with the **Help** button, press the **Actions** button () on the Settings tab of the *Screen Properties*.



The Help button options on the Settings tab of the Screen Properties dialog

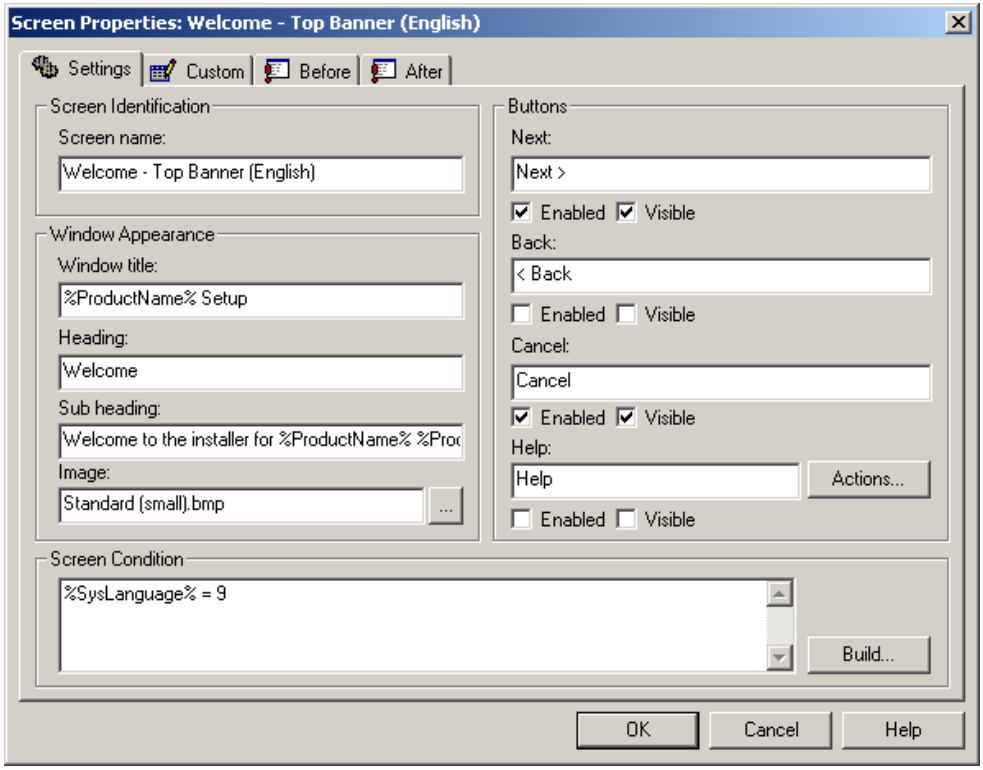
Screen Properties

The *Screen Properties* dialog is where you can configure the settings for the currently selected screen.

To access the *Screen Properties* dialog:

1. Select the screen that you want to configure on the *Screens* dialog.

2. Press the **Properties** button ( ), use the Ctrl+P hotkey, or right-click and select **Properties** from the context menu.



The Screen Properties dialog for a Welcome - Top Banner screen

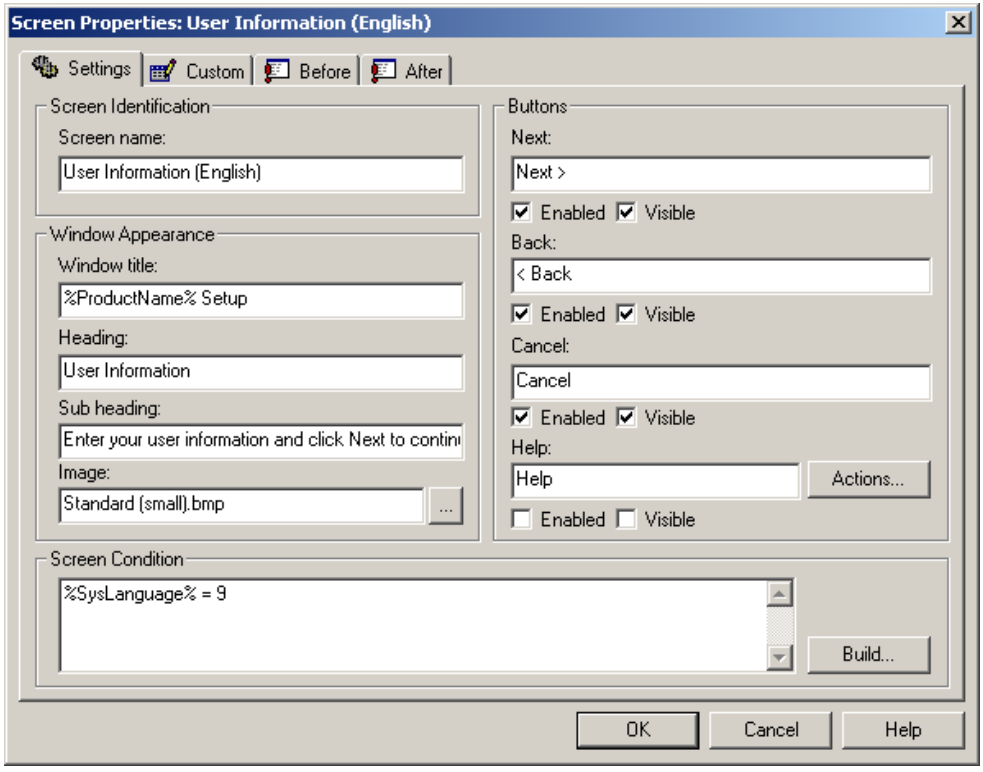
The Screen Properties dialog has four tabs: Settings, Custom, Before, and After.

Settings Tab

The Settings tab is where you can configure common screen options like the title bar text, heading text, and the text that appears on the buttons at the bottom of the screen. You can also configure which buttons will appear on the screen, and whether they will be enabled or disabled (also known as "ghosted" or "greyed out").

The Settings tab is also where you can edit the screen condition that determines whether the screen is displayed at run time.

The Settings tab displays the same options for every screen in your project.



The Settings tab for the User Information screen

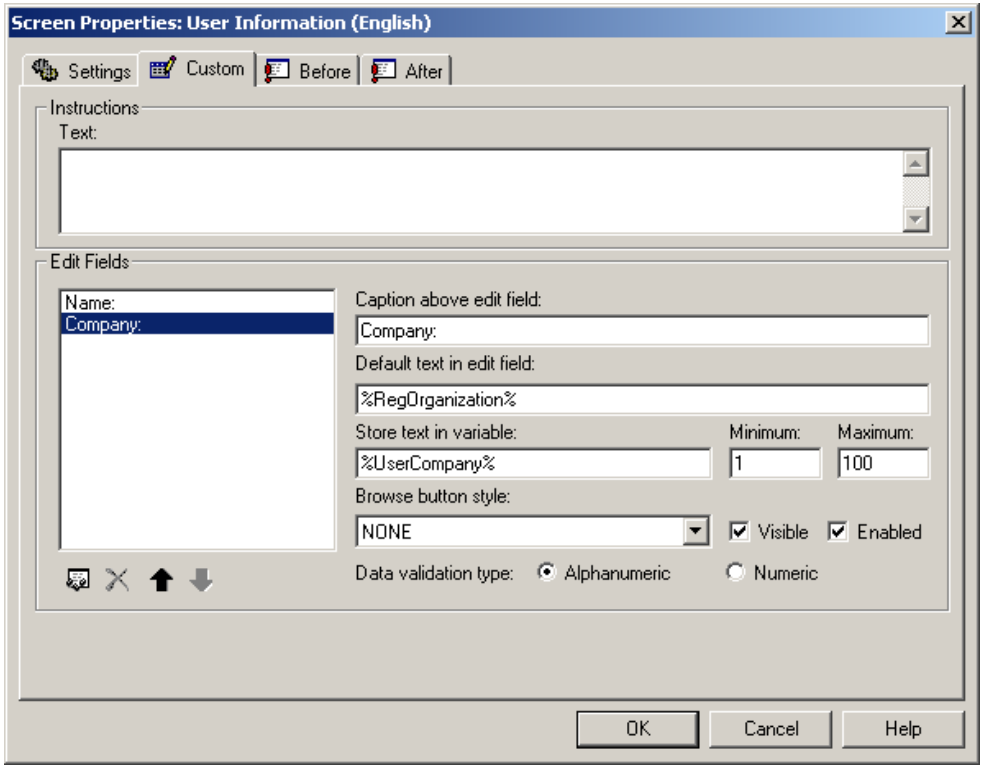
Custom Tab

The Custom tab is where you can configure screen-specific settings. This can include anything from text instructions and field captions, to default values and custom variable names.

The options on the Custom tab depend on the type of screen that is being configured.

SEE ALSO

? For more information on the custom tabs for the different screens in Setup Factory, please consult the Command Reference.



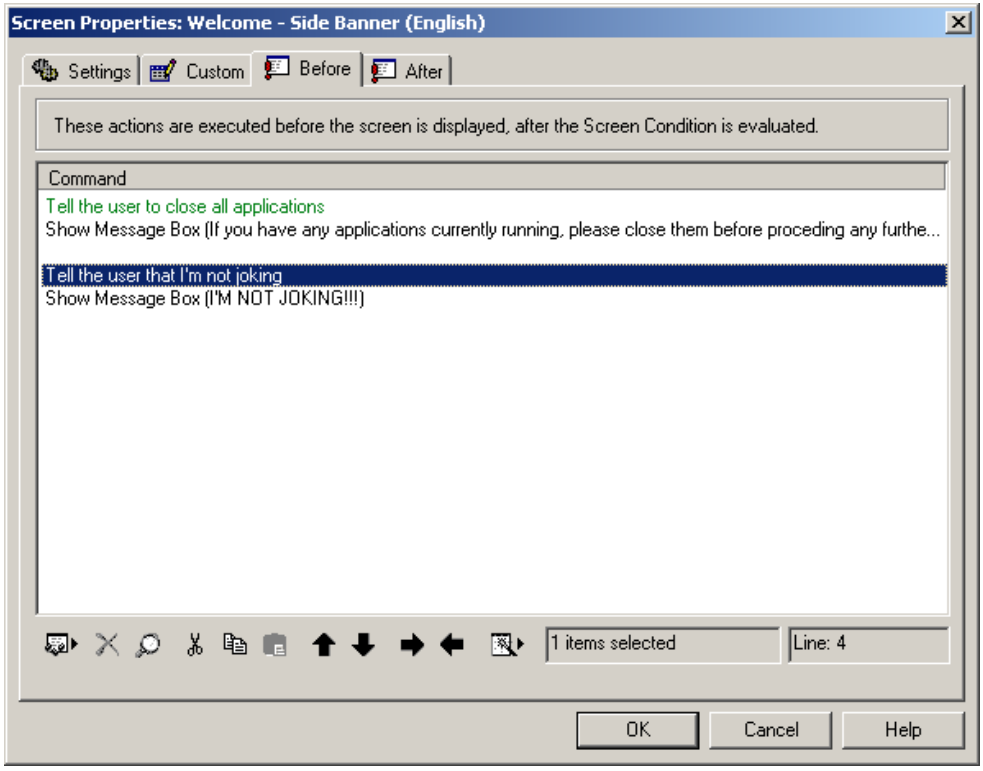
The Custom tab for the User Information screen

Before Tab

The Before tab is for actions you want performed *before* the currently selected screen is displayed. These actions are performed *after* the screen condition is evaluated, so if the screen condition isn't met, none of the actions on this tab will be performed. In other words, the actions on this tab are only performed if the screen *is* going to be displayed.

SEE ALSO

? For more information on actions, see page 159.



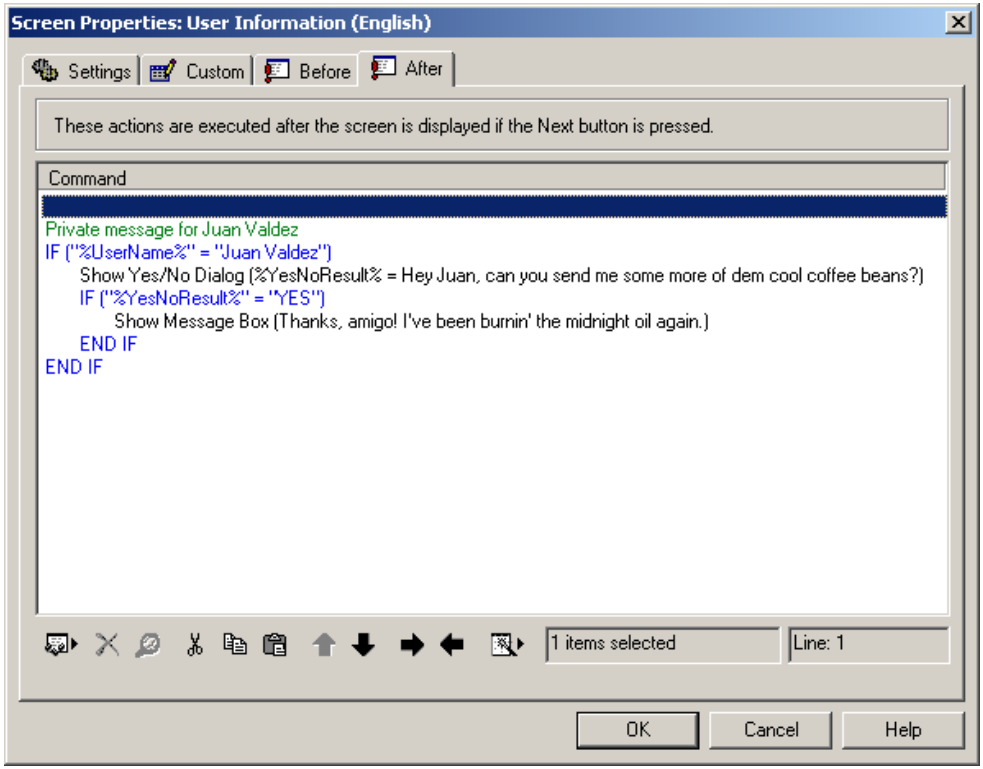
The Before tab of the Screen Properties dialog

After tab

The After tab is for actions you want performed *after* the currently selected screen is displayed. These actions are performed after the user presses the **Next** button to advance to the next screen.

SEE ALSO

? For more information on action tabs, see page 161.

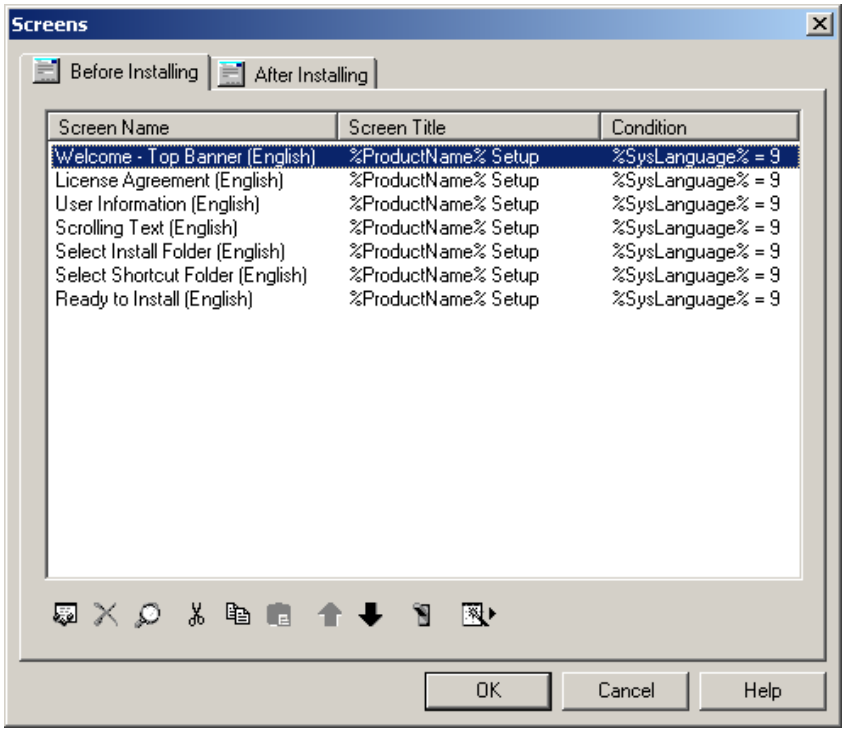


The After tab of the Screen Properties dialog

Adding Screens

To add a screen:

1. Open the *Screens* dialog by selecting **Design | Screens** from the menu.



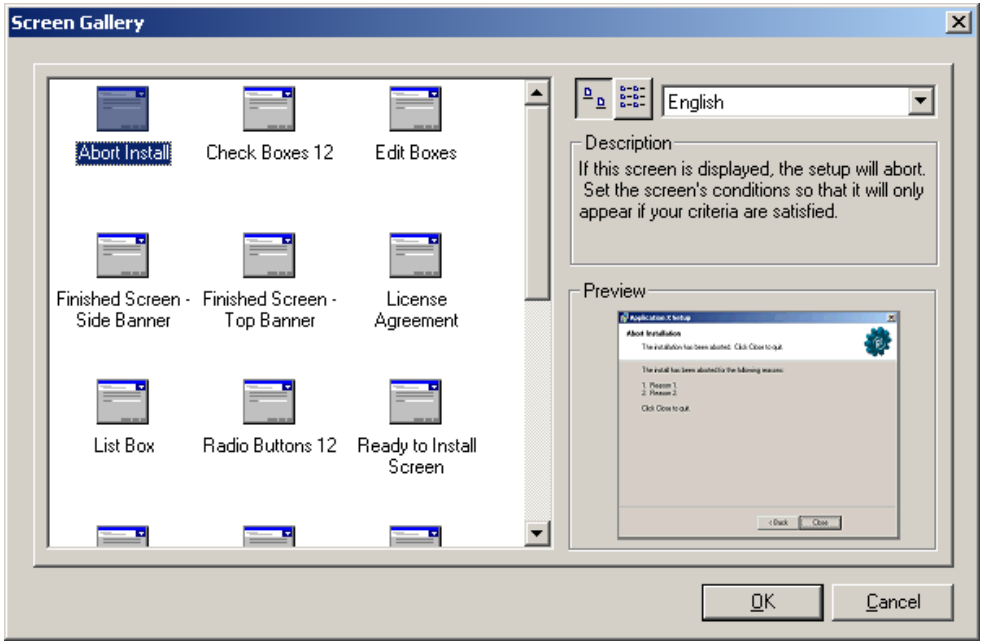
The Before Installing tab of the Screens dialog

2. If you want to add a screen that will appear *before* your files are installed, select the Before Installing tab.

   If you want to add a screen that will appear *after* your files are installed, select the After Installing tab.

3. Select the line on the *Screens* dialog where you want the screen to be inserted.

4. Press the **Add** button ( ), use the Insert hotkey, or right-click and select **Add** from the context menu. This will open the Screen Gallery.



The Screen Gallery

5. *Optional:*
   Select a language in the drop-down list on the upper right corner of the Screen Gallery. This will display the screens for that language in the list on the left.

6. Select the screen that you want from the list of screens on the left.

NOTE

A preview of the selected screen will appear on the right-hand side of the Screen Gallery.

7. Press the **OK** button to add the selected screen to the *Screens* dialog.

Removing Screens

To remove a screen:

1. Select the screen that you want to remove on the *Screens* dialog.
2. Press the **Remove** button ().

Editing Screens

To edit a screen's properties:

1. Select the screen you want to edit on the *Screens* dialog.
2. Press the **Properties** button (), use the Ctrl+P hotkey, or right-click and select **Properties** from the context menu. This will open the *Screen Properties* dialog for the screen you selected.
3. Edit the properties for the screen.

TIP

You can also double-click on a screen to display its *Screen Properties* dialog.

Rearranging Screens

Your screens will be displayed in the same order as they're listed on the Before Installing and After Installing tabs. The screens at the top of the list are displayed first, and the screens at the bottom of the list are displayed last.

To change the order that your screens will appear in:

1. Select the screen that you want to move on the *Screens* dialog.
2. Use the **Move Up** () and **Move Down** () buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition that screen in the list.

TIP

You can drag and drop individual screens to reposition them in the list. To drag and drop a screen, left-click on the screen and hold the left mouse button down while you drag the screen to another line.

TIP

You can move screens between the Before Installing and After Installing tabs by cutting and pasting them.

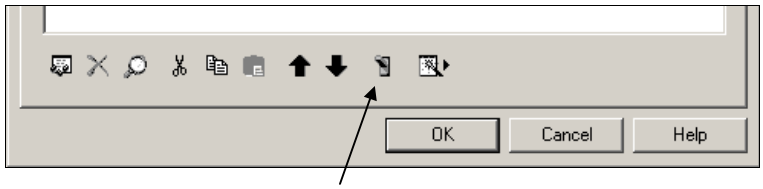
Previewing Screens

Once you've configured a screen, you can see how it will look at run time without having to build the installer.

To preview a screen:

1. Select the screen that you want to preview on the *Screens* dialog.

2. Press the **Preview** button () at the bottom of the *Screens* dialog. Setup Factory will display a preview of the screen with its current settings.



The Preview button

3. When you've finished examining your work, press any of the buttons at the bottom of the screen to return to the *Screens* dialog.

Cutting, Copying and Pasting Screens

You can cut, copy and paste screens within the same tab, between the Before Installing and After Installing tabs, and even between Setup Factory projects.

NOTE

When you cut or copy screens, they are placed in the Windows clipboard. You can open a different project file or even switch to another instance of Setup Factory to copy screens from one project to another.

To cut a screen and place it in the clipboard:

1. Select the screen you want to cut.
2. Press the **Cut** button (), use the Ctrl+X hotkey, or right-click on the screen and select **Cut** from the context menu.

To copy a screen and place it in the clipboard:

1. Select the screen you want to copy.
2. Press the **Copy** button (), use the Ctrl+C hotkey, or right-click on the screen and select **Copy** from the context menu.

To paste a screen from the clipboard onto the Before Installing or After Installing tab:

1. Select the line on the *Screens* dialog where you want the screen to be inserted.
2. Press the **Paste** button (), use the Ctrl+V hotkey, or right-click on the screen and select **Paste** from the context menu.

TIP

You can hold a Ctrl or Shift key down to select multiple screens when preparing to cut or copy them.

Exporting Screens

If you find that you're often making the same changes to a particular type of screen, you might want to consider exporting a version of the screen to use as a template. Once you've exported a screen, you can add a new copy of it right from the Screen Gallery, with all of your custom settings intact.

Exporting a screen essentially allows you to build your own screen types using the existing Setup Factory screens as templates. When you export a screen, you're saving a copy of the screen as it is currently configured. This feature is especially useful when translating screens to foreign languages.

To export a screen:

1. Select the screen that you want to export on the *Screens* dialog.

2. Press the **Advanced operations** button () at the bottom of the *Screens* dialog, and select **Export Screen** from the advanced operations menu. This will open the *Save As* dialog so you can name the data file that will contain the screen's properties.

3. Enter an appropriate name for the screen's data file.

   In order for the screen to be listed in the Screen Gallery, you must save the file beneath the `Setup Factory\Screens` folder with a `.dat` file extension.

   For example, you might save the file as:
   `C:\Program Files\Setup Factory\Screens\English\myscreen.dat`

4. Press the **Save** button to save the screen's settings to the `.dat` file.

5. *Optional:*
   Using a text editor, create an INI file for the screen in the folder where you saved the `.dat` file. This INI file is where you can specify the name and description of the screen that will appear in the Screen Gallery, as well as the names of the screen's `.dat` and `.bmp` files. (Please open one of the existing INI files to see the proper format.)

6. *Optional:*
   If you want a preview image to be displayed on the Screen Gallery when your exported screen is selected, create a preview bitmap (`.bmp`) image of your screen from a screenshot, and save it beneath the `Setup Factory\Screens` folder. (Please examine one of the existing preview bitmaps with an image editor to see the proper dimensions and image format.)

Importing Screens

To import a screen that you've exported:

1. Press the **Advanced operations** button () at the bottom of the *Screens* dialog, and select **Import .DAT File** from the advanced operations menu. This will open the *Open* dialog so you can browse for the data file that you want to import.

2. Enter the path and filename of the `.dat` file for the screen you want to import in the **File name** field, or select the screen using the *Open* dialog.

3. Press the **Open** button () to import the screen described by the `.dat` file into your project.

Chapter 10

Actions

What Are Actions?

Actions are specialized commands that your installer can perform at run time. Each action is a discrete instruction that tells the installer to do something—whether it's to open a document, search for a file, create a shortcut, or modify a Registry key.

A wide variety of actions are available in Setup Factory. In fact, there are more than fifty different actions you can use, from simple file operations to advanced functions like the Submit to Web and Read File Association actions. There are actions to register DLLs, actions to register fonts, actions to manipulate Windows services, actions to manipulate strings, actions to start and stop programs, actions to get input from the user, and more.

SEE ALSO

? For a complete list of the 50+ actions, see page 263 in this user's guide, or see the *Actions Index* in the Command Reference.

Actions are a lot like programming statements, but you don't need to be a programmer to use them. At their simplest, actions are just commands that you can use to perform various installation tasks. Need to write a value to the Registry? Add a Modify Registry action. Need to define a custom variable? Add an Assign Value action. It's really that simple.

Of course, actions can also be used together in very advanced ways. The IF, WHILE, and Assign Value actions provide the basic tools a programmer needs to build sophisticated decision-making into an installer. You can use the IF and END IF actions to form conditional blocks, and you can use the WHILE and END WHILE actions to set up loops. You can even jump between lines on an action tab using the Label and GOTO Label actions.

Actions in a Nutshell

Here's a basic overview of actions:

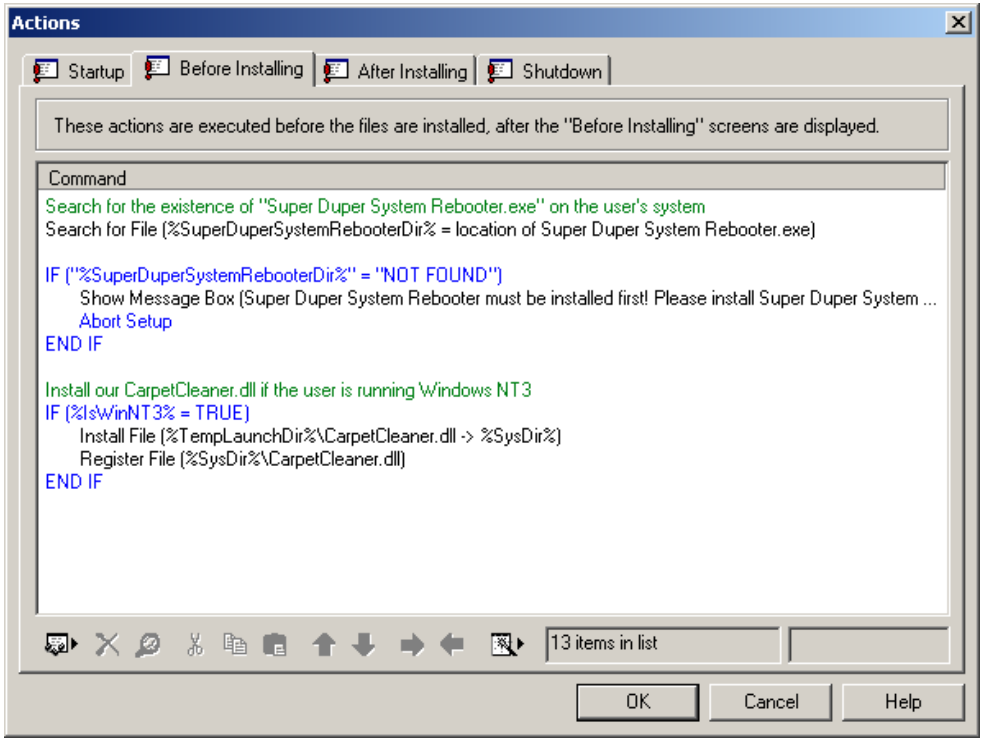
- Actions can happen at different times during the installation process; these times are represented by the various action tabs in the Setup Factory design environment.
- Actions are performed in the same order as they're listed on the action tabs (from the top down).
- You can add as many actions to your project as you want.
- You can mix and match actions to perform complex tasks.
- You can use actions to define custom variables.
- You can share information between actions by using variables.
- Actions are flexible...if there's any installation task you need performed, chances are you can do it with actions.
- You can use actions to customize the installation process completely—even replacing some of the built-in functionality of Setup Factory.
- You can import and export action lists to share them with others.
- You can use IF and END IF actions to form conditional blocks.
- You can use WHILE and END WHILE actions to set up loops.
- You can use Label and GOTO Label actions to jump between lines on an action tab.
- You can indent blocks of actions to help set them apart visually on the action tabs.
- You can use the Blank Line and Comment actions to clarify your action lists with whitespace and notes.

Action Lists

An action list is simply a sequence of one or more actions. We usually use this term to refer to a sequence of actions that performs a single function or serves a common purpose. It doesn't have to be that specific, though; "action list" can just as easily refer to all the actions on a tab, or to a handful of unrelated actions that you want to select and export to a file for future use.

Action Tabs

Action tabs are found on several dialogs throughout Setup Factory. Each action tab corresponds to a different time during the installation process when actions can be performed.



The Before Installing tab of the Actions dialog

For example, the Before Installing tab of the *Actions* dialog is where you would put all the actions that you want performed immediately before the files in your project are installed.

You can add as many actions to an action tab as you want. The actions on a tab are performed in sequence from the top down, like lines in a program. In fact, each action tab is essentially a miniature program that you can build into your installer.

There are four dialogs in Setup Factory where action tabs can be found: the *Actions* dialog, the *Screen Properties* dialog, the *Help Button Actions* dialog, and the *Uninstall* dialog.

The Actions Dialog

The *Actions* dialog has four action tabs: Startup, Before Installing, After Installing, and Shutdown.

Startup

The Startup tab is for actions you want performed at the very beginning of the installation process, before any files are installed and before any screens are displayed.

Before Installing

The Before Installing tab is for actions you want performed *before* Setup Factory begins installing the files you added to the Archive and CD-ROM tabs, *after* the "Before Installing" screens are displayed.

After Installing

The After Installing tab is for actions you want performed *after* Setup Factory finishes installing the files you added to the Archive and CD-ROM tabs, *before* the "After Installing" screens are displayed.

Shutdown

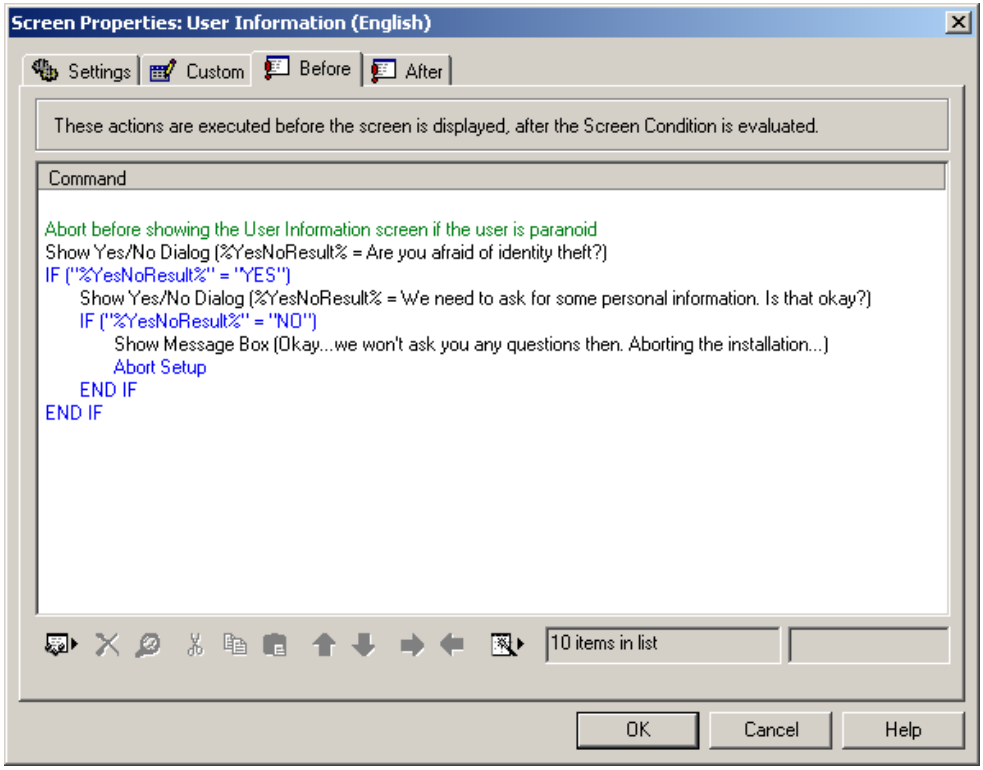
The Shutdown tab is for actions you want performed at the end of the installation process, after all the files are installed and after all the screens are displayed. In other words, this tab is for actions you want performed right before the installer exits.

The Screen Properties Dialog

The *Screen Properties* dialog has two action tabs: Before, and After.

Before

The Before tab is for actions you want performed *before* the currently selected screen is displayed. These actions are performed *after* the screen condition is evaluated, so if the screen condition isn't met, none of the actions on this tab will be performed. In other words, the actions on this tab are only performed if the screen *is* going to be displayed.



The Before tab of the Screen Properties dialog for the User Information screen

After

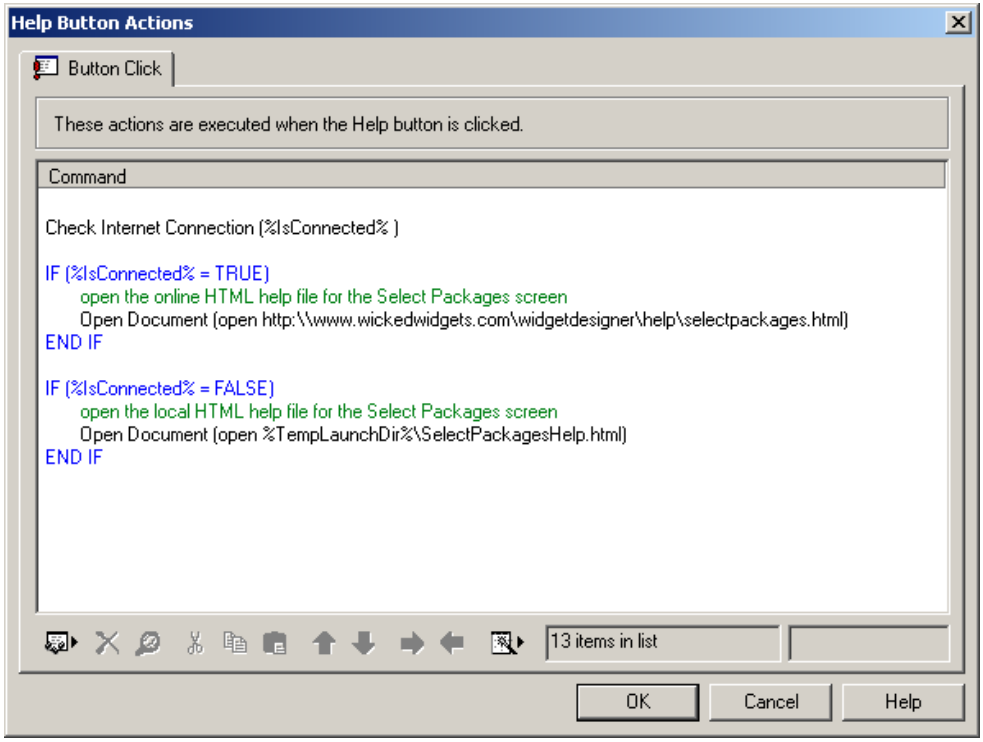
The After tab is for actions you want performed *after* the currently selected screen is displayed. These actions are performed after the user presses the **Next** button to advance to the next screen.

The Help Button Actions Dialog

The *Help Button Actions* dialog has one action tab: Button Click.

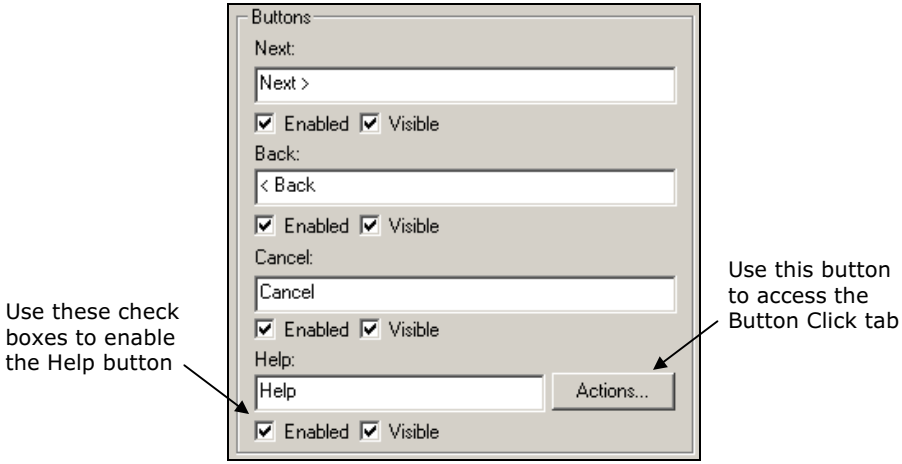
Button Click

The Button Click tab is for actions you want performed when the user presses the optional **Help** button for the currently selected screen.



The Button Click tab of the Help Button Actions dialog

To access the *Help Button Actions* dialog, press the **Actions** button (Actions...) on the Settings tab of the *Screen Properties* dialog.



The Actions button on the Settings tab of the Screen Properties dialog

TIP

You can display a **Help** button on any screen in your installer by enabling it on the Settings tab of the *Screen Properties* dialog for that screen.

To enable a button, simply select the **Enabled** and **Visible** check boxes for that button on the Settings tab.

The Uninstall Dialog

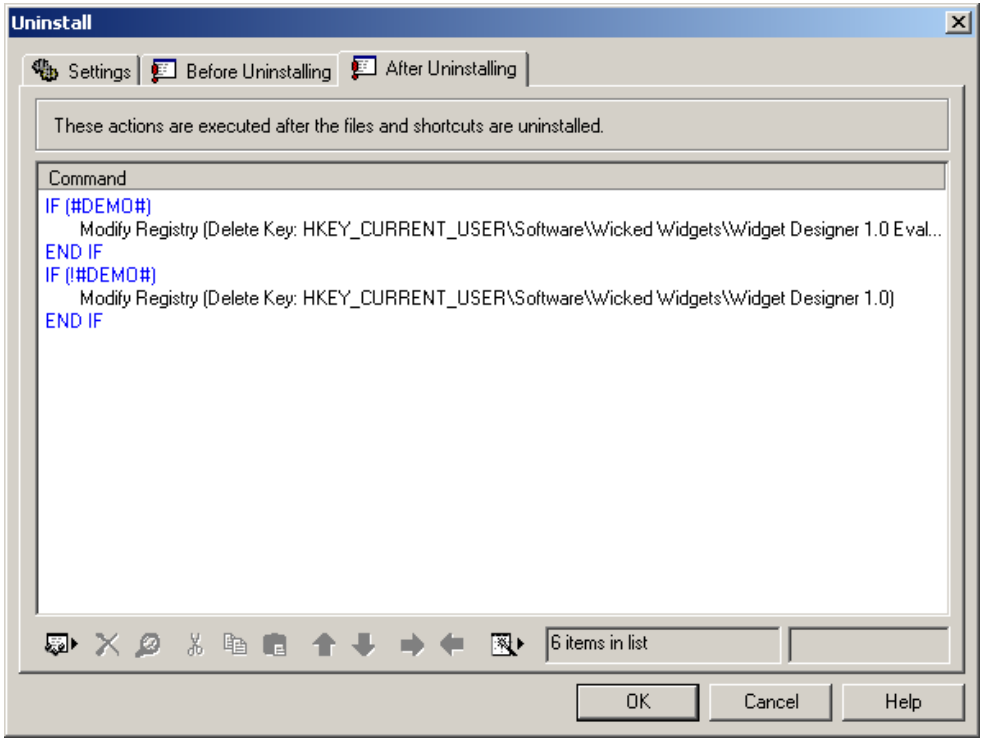
The *Uninstall* dialog has two action tabs: Before Uninstalling, and After Uninstalling.

Before Uninstalling

The Before Uninstalling tab is for actions you want performed *before* Setup Factory begins uninstalling the files you added to the Archive and CD-ROM tabs.

After Uninstalling

The After Uninstalling tab is for actions you want performed *after* Setup Factory finishes uninstalling the files you added to the Archive and CD-ROM tabs.



The After Uninstalling tab of the Uninstall dialog

NOTE

The action tabs on the *Uninstall* dialog allow you to "undo" any custom actions that were taken when your software was installed. You only need to use these tabs if your installer performs any actions at installation that aren't automatically undone by the uninstaller.

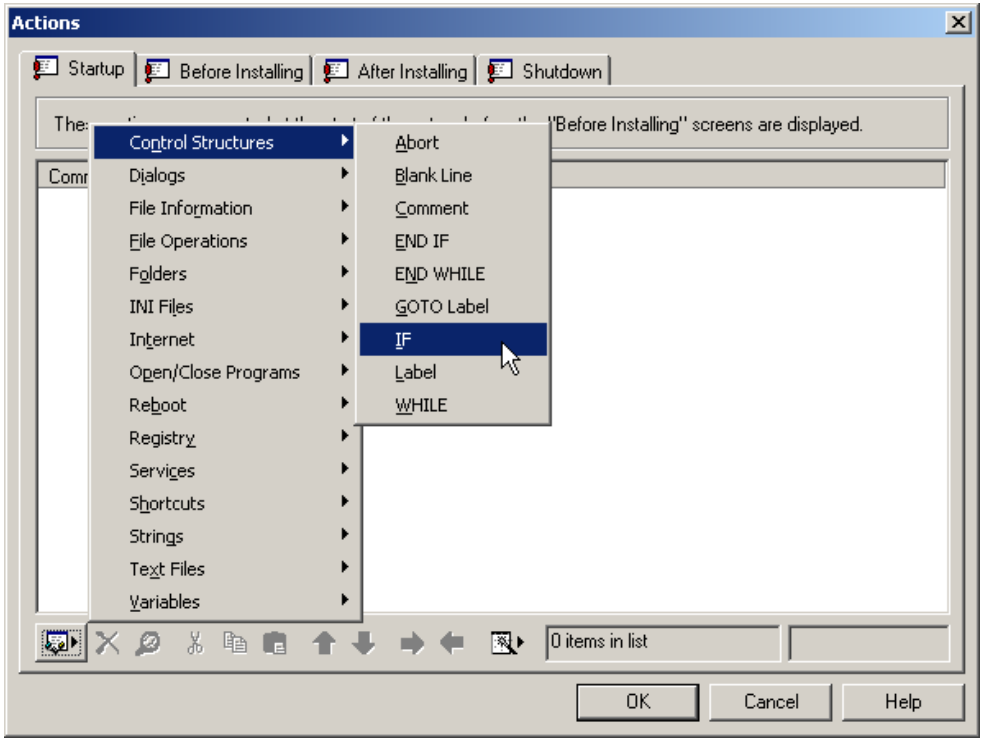
The general rule of thumb is, if you add an action to do something during the installation process, and it needs to be undone when your software is uninstalled, you'll need to add actions to the *Uninstall* dialog to do so.

Adding Actions

To add an action:

1. Select the line on the action tab where you want the action to be inserted.

2. Press the **Add Action** button ( ), use the Insert hotkey, or right-click and select **Add** from the context menu. This will open the list of action categories right next to the **Add Action** button.



The list of actions for the Control Structures category

3. Select the category that the action you want to add belongs to.

4. Select the action you want to add from the list that pops up. This will open the appropriate *Action Properties* dialog.

5. Edit the properties for the action.

SEE ALSO

? For a complete list of the actions you can add, see *Actions Index* in the Command Reference, or see page 263 in this User's Guide.

For detailed information on each action, including settings and error codes, please consult the Command Reference.

Removing Actions

To remove an action from the Actions tab:

1. Select the action you want to remove.

2. Press the **Remove Action** button ( ), use the Delete hotkey, or right-click on the action and select **Remove** from the context menu.

Editing Actions

To edit an action's properties:

1. Select the action you want to edit.

2. Press the **Properties** button ( ), use the Ctrl+P hotkey, or right-click and select **Properties** from the context menu. This will open the *Action Properties* dialog for the action you selected.

3. Edit the properties for the action.

TIP

You can also double-click on an action to display its *Action Properties* dialog.

Rearranging Actions

Actions are performed from the top down in the same order they're listed in. The action at the top of a list is performed first, and the action at the bottom of a list is performed last.

To change the order that actions are performed in:

1. Select the action you want to move.
2. Use the **Move Up** ( ) and **Move Down** ( ) buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition that action in the list.

TIP

You can drag and drop individual actions to reposition them in an action list. To drag and drop an action, left-click on the action and hold the left mouse button down while you drag the action to another line.

TIP

You can rearrange multiple actions at once by cutting and pasting them.

Indenting Actions

You can add indentation to help make your action lists easier to read. This is especially useful to help set blocks of actions apart when using actions like IF and END IF.

A good rule of thumb is to indent your code by one level after every IF or WHILE action, and to unindent it by one level at every END IF or END WHILE.

NOTE

Indentation has no effect on how the actions are performed at run time. Actions have the same effect whether they're indented or not.

To indent a block of actions:

1. Select the actions you want to indent.

2. Press the **Increase Indent** button ( ), use the Ctrl+H hotkey, or right-click on the actions and select **Increase Indent** from the context menu.

Unindenting Actions

To remove a level of indentation from a block of actions:

1. Select the actions you want to remove a level of indentation from.

2. Press the **Decrease Indent** button ( ), use the Ctrl+G hotkey, or right-click on the actions and select **Decrease Indent** from the context menu.

Cutting, Copying and Pasting Actions

You can cut, copy and paste actions within the same action tab, between action tabs, and even between Setup Factory projects.

NOTE

When you cut or copy actions, they are placed in the Windows clipboard. You can open a different project file or even switch to another instance of Setup Factory to copy actions from one project to another.

To cut an action and place it in the clipboard:

1. Select the action you want to cut.

2. Press the **Cut** button ( ), use the Ctrl+X hotkey, or right-click on the action and select **Cut** from the context menu.

To copy an action and place it in the clipboard:

1. Select the action you want to copy.

2. Press the **Copy** button (), use the Ctrl+C hotkey, or right-click on the action and select **Copy** from the context menu.

To paste an action from the clipboard onto the Actions tab:

1. Select the line on the Actions tab where you want the action to be inserted.

2. Press the **Paste** button (), use the Ctrl+V hotkey, or right-click on the action and select **Paste** from the context menu.

TIP

You can hold a Ctrl or Shift key down to select multiple actions when preparing to cut or copy them.

Exporting Actions

You can build a library of often-used actions by exporting them to Setup Factory Action Archive (`.sfa`) files. This also makes it easier to share action lists with other users.

To export actions from the Actions tab:

1. Select the actions you want to export.

2. Press the **Advanced operations** button () at the bottom of the *Actions* dialog, and select **Export Actions** from the advanced operations menu.

   (You can also use the Ctrl+E hotkey, or right-click on the actions and select **Export Actions** from the context menu.)

3. Enter a name for the Setup Factory Action Archive (`.sfa`) file on the *Save As* dialog and press the **Save** button.

Importing Actions

To import actions from a Setup Factory Action Archive (`.sfa`) file:

1. Select the line on the action tab where you want the actions to be inserted.

2. Press the **Advanced operations** button () at the bottom of the *Actions* dialog, and select **Import Actions** from the advanced operations menu.

   (You can also use the Ctrl+T hotkey, or right-click on the actions and select **Import Actions** from the context menu.)

3. Use the *Open* dialog to select the Setup Factory Action Archive (`.sfa`) file containing the actions you want to import.

4. Press the **Open** button to import the actions from the selected file.

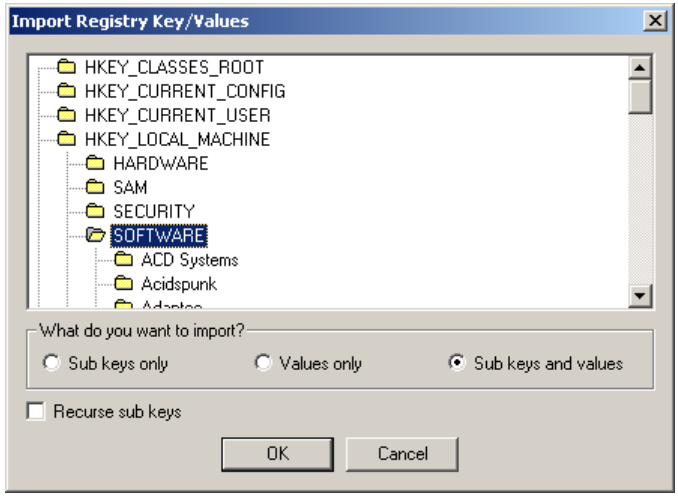
Importing Registry Values

You can "import" registry keys and values from your system's Registry and insert them into an action list in the form of Modify Registry actions. This can be especially useful when you need to make several Registry modifications for your software.

To import a registry value:

1. Select the line on the action tab where you want the Modify Registry actions to be inserted.

2. Press the **Advanced operations** button () at the bottom of the *Actions* dialog, and select **Import Registry Values** from the advanced operations menu. This will open the *Import Registry Key/Values* dialog.



The Import Registry Key/Values dialog

3. Use the *Import Registry Key/Values* dialog to select the Registry key you want to import.

4. If you want to import all of the sub keys from the key you selected, but none of the values, select the **Sub keys only** option.

   If you want to import all of the values from the key you selected, but none of the sub keys, select the **Values only** option.

   If you want to import both the sub keys and values from the key you selected, select the **Sub keys and values** option.

5. *Optional:*
   Select the **Recurse sub keys** check box if you want to import the sub keys and/or values from all of the sub keys found beneath the Registry key you selected. In other words, this check box controls whether any Registry keys or values "deeper" than the one you selected will be imported as well.

6. Press the **OK** button to import the selected Registry keys and/or values.

   Setup Factory will automatically add the appropriate Modify Registry actions to the actions list.

Using Control Structures

Setup Factory provides several actions that you can use to build control structures into your installer: IF, END IF, WHILE, END WHILE, Label, GOTO Label, and Abort.

IF and END IF

The IF action begins what's known as a *conditional block*. A conditional block is simply a series of actions that will only be performed if a condition is met. In this case, the condition is an expression entered on the *Action Properties* dialog for the IF action itself.

Each IF action must be paired with an END IF action. The END IF action marks the end of the conditional block begun by the corresponding IF action.

A conditional block is therefore defined as the series of actions between an IF action and the corresponding END IF action.

Here's an example of a simple conditional block:

```
IF (%VariableBeingTested% = 5)
    Show Message Box (The value of the variable is five!)
END IF
```

NOTE

The actions in a conditional block are usually indented to help set them apart visually from other actions.

If an IF action's condition evaluates to a true result, all of the actions between the IF action and the next corresponding END IF action are performed. If an IF action's condition

evaluates to a false result, the conditional block is "skipped," and the next action after the END IF action is performed.

For instance, in the preceding example, the message "The value of the variable is five!" would only be shown if the value in the variable %VariableBeingTested% was 5.

Conditional blocks can be nested, which is to say that you may begin and end one conditional block within another.

For example:

```
IF ("%FavoriteMusic%" = "Jazz")
    Show Message Box (Hey there cool cat.)
    IF ("%FavoriteSong%" = "PenniesFromHeaven")
        Show Message Box (Here's your favorite song.)
        Open Document (open %AppDir%\Billie Holiday\Pennies From Heaven.mp3)
    END IF
END IF
```

In this example, the first IF action matches up with the last END IF action, and the second IF action matches up with the first END IF action. The second IF action that tests %FavoriteSong% to see if it contains the string "PenniesFromHeaven" would only be performed if %FavoriteMusic% contained the string "Jazz".

WHILE and END WHILE

The WHILE action begins what's known as a *while loop*. A while loop is simply a series of actions that will be performed repeatedly for as long as a condition is met. In this case, the condition is an expression entered on the *Action Properties* dialog for the WHILE action itself.

Each WHILE action must be paired with an END WHILE action. The END WHILE action marks the end of the loop begun by the corresponding WHILE action.

A while loop is therefore defined as the series of actions between a WHILE action and the corresponding END WHILE action.

Here's an example of a short while loop:

```
Assign Value (%counter% = 1)
WHILE (%counter% < 5)
    Create Directory (%AppDir%\Folder %counter%)
    Assign Value (%counter% = %counter% + 1)
END WHILE
```

NOTE

The actions in a while loop are usually indented to help set them apart visually from other actions.

If a WHILE action's condition evaluates to a true result, all of the actions between the WHILE action and the next corresponding END WHILE action are performed. When the END WHILE action is reached, Setup Factory goes back to the corresponding WHILE action and evaluates its condition again. The actions between the WHILE and the corresponding END WHILE continue to "loop" in this fashion until the WHILE action's condition evaluates to a false result. When that happens, the actions in the while loop are "skipped," and the next action after the END WHILE action is performed.

For instance, in the preceding example, the while loop would be performed repeatedly for as long as the value in the variable %counter% was less than 5. Since %counter% is set to 1 before the while loop and incremented by 1 before the END WHILE action, this while loop would be performed four times. The result of the loop would be four directories created in %AppDir%, named:

```
Folder 1
Folder 2
Folder 3
Folder 4
```

NOTE

Each trip through a while loop is called an "iteration."

Like conditional blocks, while loops can also be nested, which is to say that you may begin and end one loop within another. For example:

```
Assign Value (%a% = 0)
WHILE (%a% < 100)
    This loops 20 times
    Assign Value (%b% = 0)
    WHILE (%b% < 5)
        This loops 5 times
        Assign Value (%b% = %b% + 1)
    END WHILE
    Assign Value (%a% = %a% + %b%)
END WHILE
Show Message Box (%a%)
```

In this example, the inner loop would be performed 5 times during each trip through the outer loop. At the end of the last iteration through the outer while loop, the variable %a% would equal 100.

Label and GOTO Label

The Label action is used to assign a name to a specific line in an action list. This name or "label" can then be used as the destination for a GOTO Label action.

The GOTO Label action can be used to "jump" directly to the line occupied by a specific label in the same action list. You can use a GOTO label action to "skip" over other actions. For example:

```
GOTO Label (Target)
Show Message Box (This action never gets performed.)
Target
```

In this example, the Show Message Box action will never be performed, because the GOTO Label action causes Setup Factory to jump directly to the label named "Target".

Although GOTO Label actions have no intelligence of their own, you can place them in conditional blocks to make jumps that are only performed when certain conditions are met. For example:

```
Check Internet Connection (%IsConnected% )
IF (%IsConnected%)
     GOTO Label (User Is Connected)
END IF

Show Message Box (You have no Internet connection!)
Abort Setup

User Is Connected
```

In this example, the GOTO Label action is only performed if the user's system is connected to the Internet.

Abort

The Abort action immediately aborts the installation process and exits the installer. This has a similar effect to the user pressing the **Cancel** button. (The **Cancel** button asks the user to confirm before exiting; the Abort action doesn't ask, but instead just exits immediately.)

An example of where you might want to use an Abort action is in a custom error handler, after an irrecoverable error has been encountered.

Adding Comments and Whitespace

Setup Factory provides two important actions that you can use to make your action lists easier to read. The first and most important one is the Comment action.

Comments

The Comment action allows you to add notes to your action lists in order to make them easier to understand. Each comment contains a line of text that will be completely ignored by the installer at run time. Because this text will be ignored, you can write anything you want in a Comment action, and it will have no effect on how the installer operates.

Comment actions are displayed in a color that makes them easy to recognize on the action tabs. (The comments are displayed in green by default, but you can change this color on the Action Tabs tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.)

You should make a habit of adding comments to your action lists as you build them. If adding comments seems like a waste of time, consider how much time you will spend working on these actions in the future. Complex action lists that make perfect sense to you now might take a few moments to interpret the next time you need to make changes to a project. A few comments written while things are fresh in your mind can minimize the amount of re-thinking required.

Comments are especially important in a multi-user environment. If it's foreseeable at all that anyone else may need to work on your Setup Factory project in the future, do your organization a favor and document your actions well.

Here are some tips for using Comment actions:

- Use comments to summarize large blocks of actions. This way, you only have to read the comment to know what the actions do. It's easier to read one line of text than it is to decipher a long list of actions.

- Use comments to explain why an action is needed. This is especially important when the purpose of an action isn't obvious.

- Use comments to highlight important actions, or to label different parts of your action lists so they're easier to find.

- Use comments in a team environment to help keep track of changes (and who made them).

- Use comments to document the decision process that led you to choose one installation approach over another. This could save you time if you ever find yourself considering other approaches again.

- Use comments appropriately. Don't waste time explaining simple actions that are readily apparent.

Think of comments as investing a little bit of time now to save you a lot of time over the long run. You only have to write a comment once, but you will benefit from the comment many times over. In the long run, well-written comments will save you time, time and time again.

Blank Lines (Whitespace)

Blank Line actions allow you to add vertical whitespace to your action lists. ("Whitespace" is a term programmers use to describe things like blank lines and spaces that are used to improve the readability of their code.) Blank Line actions are ignored by the installer; their only purpose is to allow you to separate actions with whitespace on an action tab.

You can use Blank Line actions to make your action lists easier to read. For example, separating groups of actions with one or more blank lines makes it easier to recognize that the actions are meant to work together as a group.

Handling Errors

Even the most well-designed installers can run into situations that generate errors. For instance, connection problems might prevent an HTTP download from succeeding, or an INI file you need to read from may have been deleted by a careless user.

Setup Factory gives you two ways to respond to such errors:

Built-in error handling (The On Error tab)

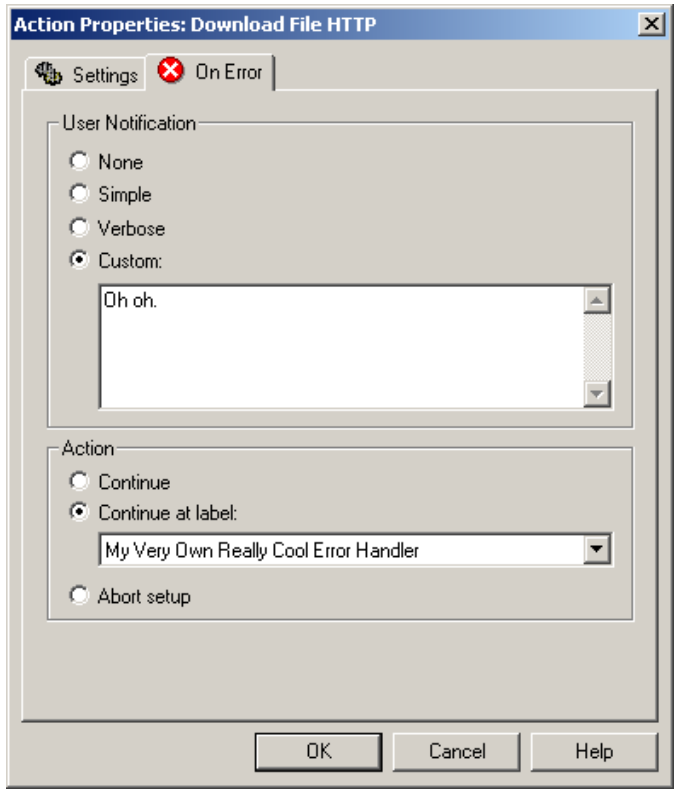
You can use the standard error handling features built into each action, which you can configure using the On Error tab of the action's properties dialog.

Custom error handling (Using actions)

You can use your own combination of actions to respond to the error. These actions could be contained in an IF/END IF block that checks built-in variables like %LastErrorNum% to determine whether an error occurred. Or, the actions could be assigned a label and called directly using the **Continue at label** setting on the On Error tab.

Built-in Error Handling (The On Error Tab)

Every action in Setup Factory that could generate an error at run time has an On Error tab on its *Action Properties* dialog. You can use the On Error tab to configure how Setup Factory responds when an error is generated by that action.



The On Error tab of an Action Properties dialog

Setting the User Notification Options

The User Notification section of the On Error tab allows you to control how much information is given to the user when an error occurs—or even whether the user is notified at all.

There are four user notification options you can choose from:

None

No error message is shown to the user; the user is not notified about the error at all.

Simple

A simple error message is displayed on a dialog window. The error message lets the user know that something happened, but provides only basic information about the type of error that occurred.

Verbose

A more specific error message is displayed. The message might include pertinent details about the error, such as the URL that a failed HTTP download action was trying to connect to.

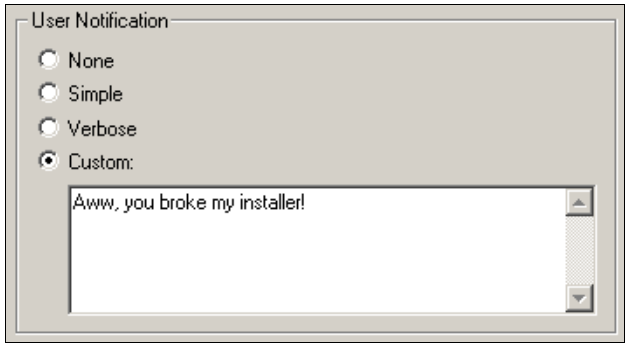
Custom

A custom error message of your own is displayed. You can enter your own message text directly onto the On Error tab when this option is selected.

TIP

You can use built-in variables like %LastErrorNum%, %LastErrorMsg% and %LastErrorDetails% to incorporate the standard error message text in your custom error messages.

To set the level of user notification you want, simply select the appropriate option in the User Notification section of the On Error tab.



The User Notification section of the On Error tab

Setting the Action Taken After an Error Occurs

You can choose whether you want the installer to abort because of an error, or continue performing actions. If you have any labels defined on the Actions tab, you can even tell Setup Factory to go to one of the labels and continue the installation from there.

There are three on-error action settings you can choose from:

Continue

After an error, the installation continues with the next action in the list—i.e., the next action on the current action tab.

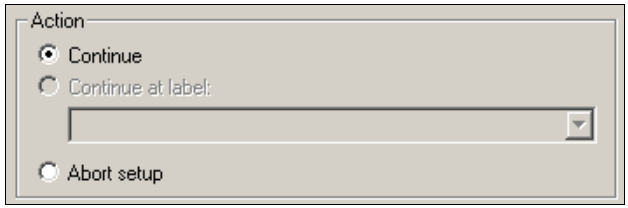
Continue at label

After an error, the installer goes directly to the specified label and continues the installation process from there. This option is only available if there is at least one label defined on the current action tab.

Abort setup

After an error, the installation is aborted. No more actions are performed, and no more screens are shown.

To set the on-error action you want, simply select the appropriate option in the Action section of the On Error tab.



The Action section of the On Error tab

Custom Error Handling (Using Actions)

In addition to the "built-in" error handling provided by Setup Factory, you can use Setup Factory actions to respond to errors in a custom fashion. For example, you might want to try an alternative download site if an HTTP download fails. Or, you might want to present a Yes/No dialog to the user and let *them* choose whether to abort the installation.

There are two ways you can trigger a custom error handling routine:

Checking %LastErrorNum%

You can use an IF action to check the built-in variable %LastErrorNum% and determine whether the previous action generated an error.

Using Continue at label

You can use the **Continue at label** option on an action's On Error tab to jump to your custom error-handling routine when that action generates an error.

Checking %LastErrorNum%

%LastErrorNum% is a built-in variable that gets updated every time an action is performed by the installer. If an action is successful, %LastErrorNum% is set to 0. If an action generates an error, %LastErrorNum% is set to a positive integer value that represents the error that occurred.

Each action has its own set of possible error numbers or "return values" that are assigned to %LastErrorNum% when an error occurs. Some actions, like Execute File, only have one error number. Others, like Download HTTP, have several.

SEE ALSO

? You can find a complete list of return values for each action in the Command Reference.

You can use %LastErrorNum% in the **Condition** field of an IF action to test whether the previous action generated an error. Any actions between the IF action and the next corresponding END IF action will only be performed when the preceding action fails.

Your error handler might look something like this:

```
Tricky download
action that might fail:
HTTP Download (http://www.unreliable.com/slim/chance.exe -> %SrcDir%)
IF (%LastErrorNum%)
    these actions are only performed when %LastErrorNum% is not 0
    Show Yes/No Dialog (%YesNoResult% = The download failed...would you like to try again?)
    IF (%YesNoResult%)
        GOTO Label (Tricky download)
    END IF
END IF
```

If you only wanted to handle specific errors, you could set things up like this instead:

```
IF ((%LastErrorNum% = 3) OR (%LastErrorNum% = 4))
    these actions are only performed when %LastErrorNum% is set to 3 or 4...
END IF
```

Three other built-in variables are updated every time an action is performed by the installer: %LastCommand%, %LastErrorMsg% and %LastErrorDetails%.

%LastCommand% is set to the ID of the action.
%LastErrorMsg% is set to the standard "simple" error message for the action.
%LastErrorDetails% is set to the standard "verbose" error message for the action.
(Both error messages are localized—they're taken from the language file.)

Note: %LastErrorMsg% and %LastErrorDetails% are only set when the previous action generated an error. Both variables are empty ("") when the previous action was successful.

Using Continue at label

Another way to trigger a custom error-handling routine is to use the **Continue at label** option found on an action's On Error tab. Just group your error-handling actions together, preface them with a label, and select that label for the **Continue at label** option.

You'll probably also want to use GOTO actions and additional labels to skip over your error-handling routines, so they remain isolated from the rest of your actions.

Using the **Continue at label** option, your error handler might look something like this:

```
normal actions above this point...
GOTO Label (Continue)
Error Handler
    these actions are only performed
    when an action fails
    and its Continue at label option
    is set to "Error Handler"
Continue
back to normal actions again...
```

TIP

You can use a variable like %LastCommand% in the **Continue at label** field to jump to labels that you've named based on the different actions whose errors you want to handle.

Chapter 11

Packages

What Are Packages?

Packages are special categories that you can define in order to group related files together. They're usually used to give users the ability to choose which parts of an application they want to install.

SEE ALSO

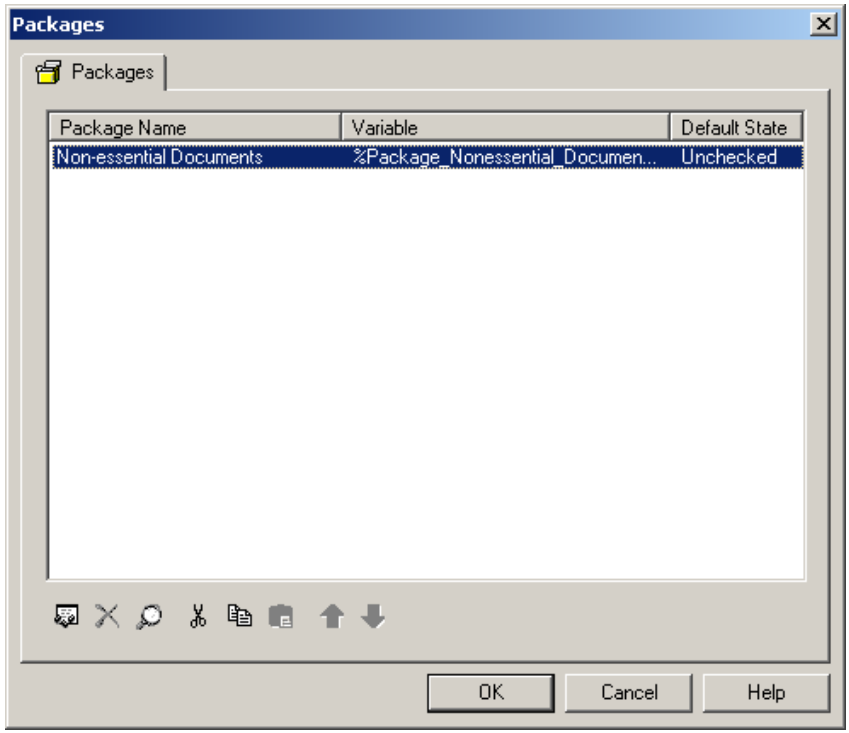
? For more information on packages, see page 41.

Packages in a Nutshell

Here's a basic overview of packages:

- Each package consists of a name, a description and a unique custom variable.
- You can assign files to packages by using the *File Properties* dialog.
- You can have as many packages in a Setup Factory project as you want.
- You can assign as many files to a package as you want.
- You can assign the same file to more than one package.
- You can associate a file with more than one package.
- You can let your users select packages by adding *Select Packages* or *Select Install Type* screens to your installer.
- Selecting a package sets its variable to "true". Deselecting a package sets its variable to "false".

- The package variable is what determines whether files are installed.
- Files assigned to a package are installed when that package's variable is set to "true".
- Files assigned to multiple packages are installed when at least one of the packages' variables is set to "true".
- Setup Factory automatically calculates the total size of the files in each package. These package sizes can optionally be displayed on the *Select Packages* screen.
- The name and description of a package can be localized so they will appear in the user's chosen language.



The Packages dialog

Using Packages

Using packages in your Setup Factory project involves three simple steps:

1. Create the packages.
2. Assign files to the packages.
3. Add screens to your project so users can select the packages.

For example, let's say you include a set of tutorials with your application in the form of very large video files. Because these files are so large, you want your users to be able to choose whether or not to install them at run time. Here's how you could accomplish this by using packages:

First, you would use the *Packages* dialog to create a package for the video files. You would give the package an appropriate name like "Video Tutorial Files" and change the default variable name to something unique and easy to remember; a good name for this package's variable would be something like %Package_VideoTutorialFiles%.

You could also write a brief description to help your users decide whether they want to install the tutorial files, e.g. "Optional tutorial files in AVI video format. Windows MediaPlayer 6 or above is required in order to view these files."

Next, you would assign the video files to the package. To do this, you would select all the video files on the project window, press the **File Properties** button () to access the *Multiple File Properties* dialog, and use the Packages tab to assign the files to the "Video Tutorial Files" package. Once this is done, the video files will only be installed if the value in %Package_VideoTutorialFiles% is set to "true".

Finally, you would use the *Screens* dialog to add a *Select Packages* screen to your project. The *Select Packages* screen is able to display any number of packages in a list, with check boxes that the user can use to turn the individual packages on or off. In order for the "Video Tutorial Files" package to appear on this screen, you need to add it to the list of packages that the screen will display. To do this, you would access the Custom tab of the *Screen Properties* dialog for the *Select Packages* screen, and add the "Video Tutorial Files" package to the list of available packages.

NOTE

By default, the *Select Packages* screen will only be displayed if the "Custom" option is chosen on a preceding *Select Install Type* screen at run time. If you aren't using a *Select Install Type* screen, you will need to modify the screen condition for the *Select Packages* screen to remove this restriction. To do so, simply edit the expression in the **Condition** field on the Settings tab of the *Screen Properties* dialog for the *Select Packages* screen.

Naming Package Variables

Because each package requires a unique custom variable name, it's a good idea to establish a naming convention for your package variables. (A naming convention is a set of rules that you follow when forming a name for something.) We recommend that you use the following naming convention for your package variables:

%<prefix><package name>%

In other words, begin each package variable with a prefix like %Package or %Package_ followed by the name of the package and a percentage sign. If the package name contains spaces, either remove them or replace them with underscores (_).

For example, if the package name is "Extra Help Files", a good name for the package variable would be something like %PackageExtraHelpFiles% or %Package_Extra_Help_Files%.

Forming a variable name based on the name of the package not only makes it easier to come up with appropriate names, it makes it easier to determine what variable name was used for an existing package.

NOTE

Using numbers to make variable names unique isn't recommended; it's difficult to know at a glance what package a variable name like %Package1234% represents.

Whichever naming convention you adopt, the most important thing is to name your package variables consistently throughout your project.

SEE ALSO

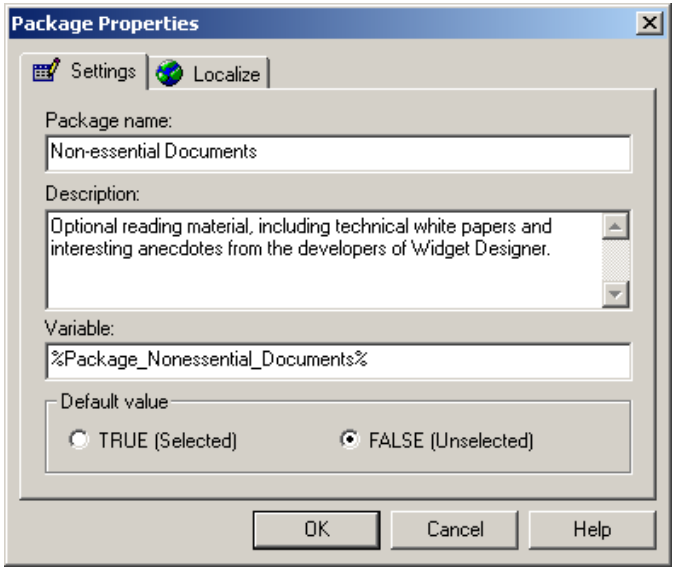
? For more information on naming variables, see page 216.

Adding Packages

To add a package to your project:

1. Select the line on the *Packages* dialog where you want the package to be inserted.

2. Press the **Add Package** button ( ), use the Insert hotkey, or right-click and select **Add** from the context menu. This will open the *Package Properties* dialog for the package you're adding.



The Package Properties dialog

3. Provide a name for the package in the **Package name** field. This is the name that your users will see on a *Select Packages* screen at run time.

4. Provide a description for the package in the **Description** field. This description will be displayed on the *Select Packages* screen when users select this package.

5. Enter a *unique* variable name for this package in the **Variable** field. This is the variable that will be set to true or false in order to enable or disable this package.

6. Select the value that you want the package to have by default, i.e. either TRUE (the package is selected) or FALSE (the package is unselected).

   This sets the initial value of the package variable, and determines whether the package will be installed if no overriding selections or assignments are made at run time.

7. *Optional:*
   Use the Localize tab to translate the package name and description into different languages.

SEE ALSO

? For more information on translating packages, see page 245.

Removing Packages

To remove a package:

1. Select the package you want to remove on the *Packages* dialog.

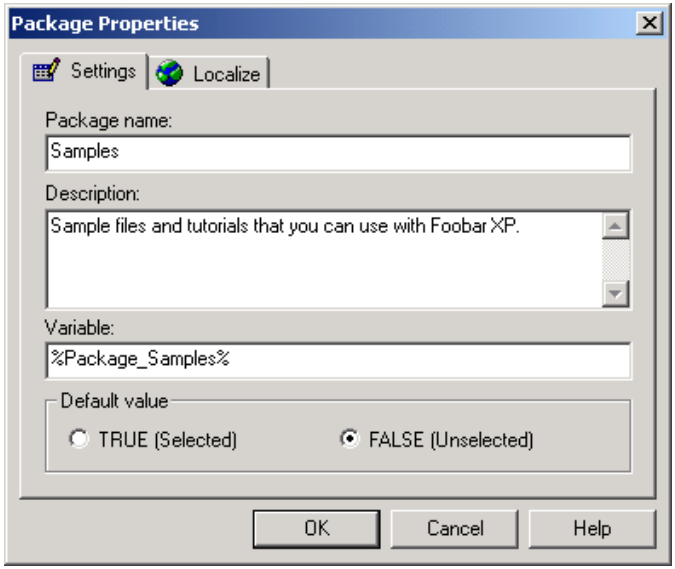
2. Press the **Remove Package** button (), use the Delete hotkey, or right-click on the package and select **Remove** from the context menu.

Editing Packages

To edit a package:

1. Select the package you want to edit on the *Packages* dialog.

2. Press the **View Properties** button (), use the Ctrl+P hotkey, or right-click and select **Properties** from the context menu. This will open the *Package Properties* dialog for the package you selected.



The Package Properties dialog

3. Edit the properties for the package.

TIP

You can also double-click on a package to display its *Package Properties* dialog.

Rearranging Packages

The order of the items on the *Packages* dialog determines the order that packages are displayed in throughout the Setup Factory design environment. By rearranging the items on the *Packages* dialog, you can control the order that packages will be listed in on the other dialogs you use at design time.

For example, the Packages tab of the *File Properties* dialog lists all of the packages in your project. You can change the order of the packages in this list by rearranging the items on the *Packages* dialog.

NOTE

This only affects the package lists that you use at design time. It has no effect on the order that packages are displayed in at run time.

To change the order of the package lists within the design environment:

1. Select the package you want to move on the *Packages* dialog.
2. Use the **Move Up** () and **Move Down** () buttons or the Ctrl+Up and Ctrl+Down hotkeys to reposition that package in the list.

TIP

To change the order that packages are displayed in at run time, edit the **Packages available on this screen** list directly in the properties of the *Select Packages* screen.

TIP

You can move multiple items at once by cutting them into the clipboard and then pasting them back onto the list at another location.

To select multiple items, hold a Ctrl or Shift key down while making your selections with the mouse.

Cutting, Copying and Pasting Packages

You can cut, copy and paste packages within the Packages tab, and even between Setup Factory projects.

NOTE

When you cut or copy packages, they are placed in the Windows clipboard. You can open a different project file or even switch to another instance of Setup Factory to copy packages from one project to another.

To cut a package and place it in the clipboard:

1. Select the package you want to cut.
2. Press the **Cut** button (), use the Ctrl+X hotkey, or right-click on the package and select **Cut** from the context menu.

To copy a package and place it in the clipboard:

1. Select the package you want to copy.
2. Press the **Copy** button (), use the Ctrl+C hotkey, or right-click on the package and select **Copy** from the context menu.

To paste a package from the clipboard onto the Packages tab:

1. Select the line on the Packages tab where you want the package to be inserted.
2. Press the **Paste** button (), use the Ctrl+V hotkey, or right-click on the screen and select **Paste** from the context menu.

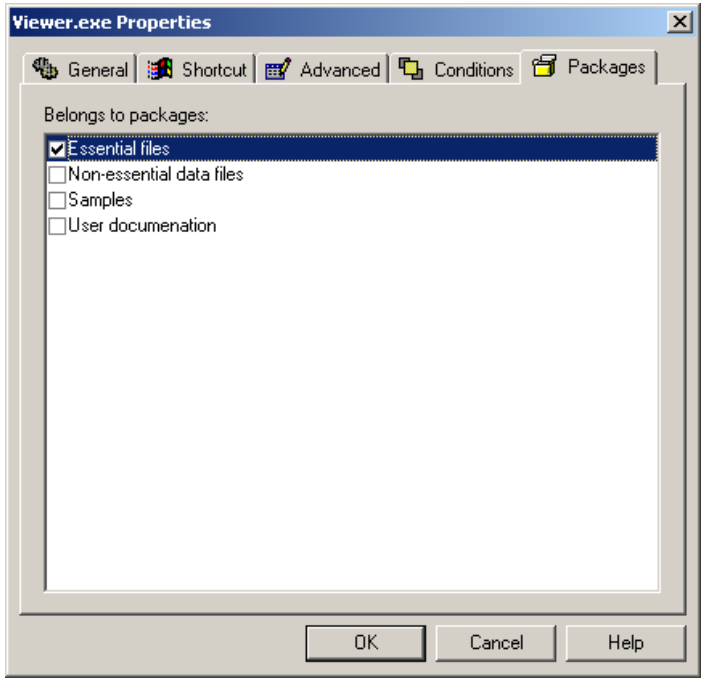
   If you're pasting a copy of a package that already exists in the list, the name and variable of the new package will be adjusted in order to prevent duplicates.

Assigning Files to Packages

To assign a file to one or more packages:

1. Select the file that you want to assign to a package on the project window.

2. Access the *File Properties* dialog for this file by pressing the **File Properties** button ( ), using the Ctrl+P hotkey, or right-clicking and selecting **File Properties** from the context menu.



The Packages tab of the File Properties dialog

3. On the Packages tab, select the packages that you want this file to belong to.

TIP

You can assign more than one file to a package at once by selecting multiple files on the project window.

Install Types

Not all of your users will want or need the degree of control over their installations that Setup Factory packages can provide them. And, true enough, some users should probably be protected from such advanced features. For this reason, many installers are designed to make packages optional, and simplify the user's decision by presenting streamlined choices in the form of *install types*.

There are four standard install types traditionally provided by installers:

Typical

A Typical install type should install a balanced set of features—the options you believe most users will want or need. Although it varies with the software you're installing, a Typical installation generally includes most of the files in your project. However, a Typical installation might exclude, for example, special drivers that only a limited number of users need, extra sample files, and extra documentation written for software developers (SDKs).

Complete

A Complete install type should install everything.

Minimum

A Minimum install type should only install files that are absolutely essential. This "bare minimum" installation would general be used when installing your software to a computer with a minimum of disk space (e.g. a laptop).

Custom

A Custom install type should allow the user to choose which individual packages are installed. Custom installations are most often used by "power users" who know what they are doing and want full control over what gets installed on their system.

In Setup Factory, install types are handled entirely by the *Select Install Type* screen. Each *Select Install Type* screen supports up to four install types. Adding, removing, configuring, and naming install types are all done directly on the Custom tab of the *Screen Properties* dialog for the *Select Install Type* screen.

You can think of install types as package "presets." Each install type will automatically enable or disable specific packages in your project, turning them individually on or off according to the *Select Install Type* screen settings.

When the user selects an install type on a *Select Install Type* screen, the name of the selected install type is assigned to a custom variable. You can use this variable in screen conditions to make some screens only appear if a specific install type was selected.

For example, the *Select Packages* screen has the following screen condition by default:

```
"%InstallType%" = "Custom"
```

This screen condition effectively hides the *Select Packages* screen unless the user selects the "Custom" install type on the *Select Install Type* screen. So, by default, the "Custom" install type appears to lead to a screen with more advanced customization options.

Packages and run-time conditions
Assigning a file to a package is just like setting a run-time condition for the file. In fact, you could completely duplicate the functionality of packages by using equivalent run-time conditions.

For example, if you have a package called "Extra Stuff" and it uses a package variable named %PackageExtraStuff%, assigning a file to that package is a lot like adding the following run-time condition to the file:

%PackageExtraStuff% = TRUE

Of course, run-time conditions don't have easily localized names and descriptions like packages do. Assigning a run-time condition to a file also doesn't add that file's size to the package size shown on the *Select Packages* screen at run time.

If you ever want to assign a file to a package without having the file's size affect the package size, set up a run-time condition instead. That way, the file will still only be installed if the package variable is set to true, but its size won't affect the package size shown on the *Select Packages* screen.

Chapter 12

Runtime Support

Some executables in your project may require other files in order to work properly. These required files are known as *runtime support files* or *dependency files*.

Dependency files are external support files that an executable requires for proper operation. In other words, they are the external files that a program file "depends on" in order to function properly. Dependency files may include INI files, DLLs, ActiveX controls, OCX components, or any other support file type.

NOTE

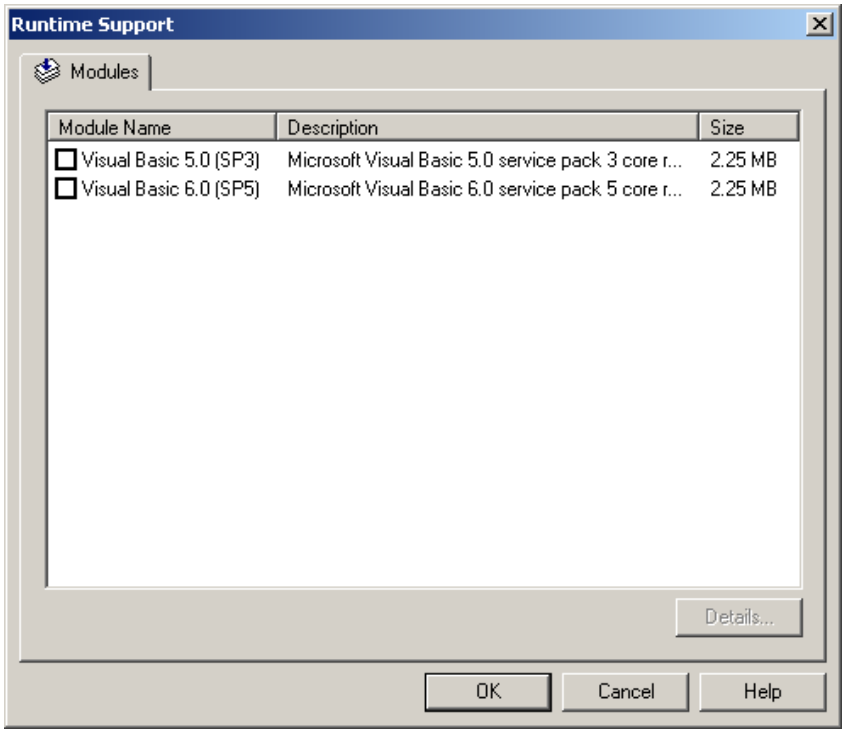
Many of today's development tools require that you distribute runtime support files along with your application. Please consult your development tool's documentation to determine what files you need to distribute with your software.

There are three ways that you can add dependency files to your installer:

- You can add pre-packaged runtime modules using the *Runtime Support* dialog.
- You can scan a Visual Basic project file for dependencies using Setup Factory's Visual Basic project scanner.
- You can scan an executable for dependencies using Setup Factory's dependency file scanner.

The Runtime Support Dialog

The *Runtime Support* dialog allows you to add pre-packaged runtime support modules to your Setup Factory project. You can access the *Runtime Support* dialog by selecting **Design | Runtime Support** from the menu, or by clicking on the Runtime Support icon on the shortcut bar.



The Runtime Support dialog

The *Runtime Support* dialog lists all of the runtime modules that can be included in your project. You can add runtime modules to your project by clicking on the check boxes in the list. Any modules that have check marks beside them will automatically be included in the setup executable when you build the current project.

Clicking on an empty check box adds a check mark, and clicking on a box that already has a check mark clears it.

An empty check box looks like this: ☐ Visual Basic 6.0 (SP5)

A selected check box looks like this: ☑ Visual Basic 6.0 (SP5)

When you select a runtime module for inclusion in your project, you won't notice any immediate changes to your project file. Instead, the necessary files and actions

will automatically be merged with the project file at build time.

To learn more about a particular runtime module, select it in the list and press the **Details** button (Details...) on the *Runtime Support* dialog. This will open any documentation available for this runtime module.

TIP

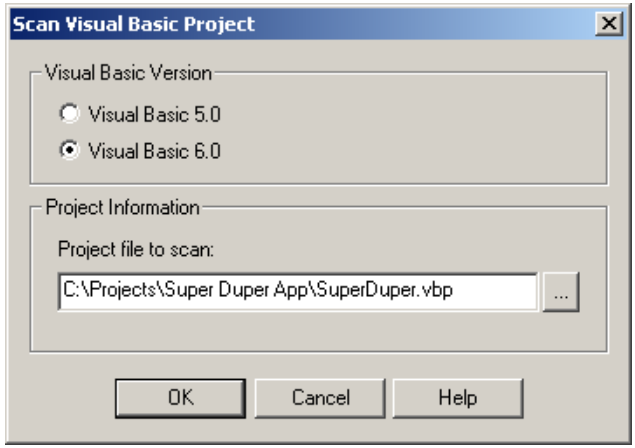
If you need a runtime module that doesn't ship with Setup Factory, check the Indigo Rose web site. We will be posting new runtime modules as they become available.

The Visual Basic Project Scanner

You can use Setup Factory's Visual Basic project scanner to scan a Visual Basic project (.vbp) file for any "extra" dependencies not found in the core Visual Basic runtime modules.

To scan a Visual Basic project file for dependencies:

1. Select **Tools | Scan Visual Basic Project** from the menu. This will open the *Scan Visual Basic Project* dialog.



The Scan Visual Basic Project dialog

2. Select the version (5.0 or 6.0) of the Visual Basic project you want to scan.

3. Enter the path and filename of the .vbp file you want to scan in the **Project file to scan** field, or press the **Browse** button () to browse for a file.

4. Press the **OK** button to begin the dependency scan.

   The Visual Basic project scanner will analyze your Visual Basic project (.vbp) file and determine any dependencies that it has beyond the core Visual Basic runtime files. When it finds other dependencies, it will attempt to determine the dependencies of those files as well. Setup Factory will then add all the dependency files to your Setup Factory project.

   If your Visual Basic project does not use any files beyond the core runtime files, no files will be added to your Setup Factory project. Instead, the appropriate Visual Basic runtime module will automatically be checked for you on the *Runtime Support* dialog.

IMPORTANT

! The Visual Basic project scanner will not add your actual executable file, or any other support files (such as help files, images, documents, etc.) other than DLL and OCX files.

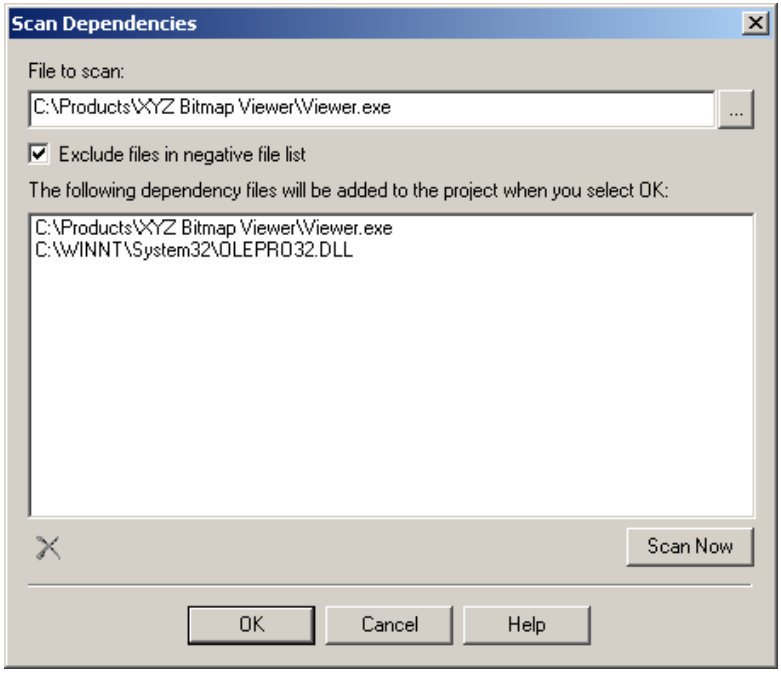
Also, bear in mind that the project importer relies on third-party control manufacturers having adhered to Microsoft's standard .DEP file format. If you are using complex third-party runtimes, you should always consult the documentation of those runtimes for more distribution information.

The Dependency File Scanner

You can use Setup Factory's dependency file scanner to scan any Portable Executable file for dependencies.

To scan a file for dependencies:

1. Select **Tools | Scan File Dependencies** from the menu. This will open the *Scan Dependencies* dialog.



The Scan Dependencies dialog

2. Enter the path and filename of the Portable Executable file that you want to scan in the **File to scan** field, or press the **Browse** button (...) to browse for a file.

3. Select the **Exclude files in negative file** list check box if you want to exclude standard system files that normally should not be distributed with your installer. (This setting is enabled by default.)

   In general, you should leave this setting enabled to avoid shipping core system files with your setup. Shipping certain system files could result in corruption of your user's system.

NOTE

The negative file list is located in the `Data` sub-folder of the Setup Factory application directory on your hard drive. It is stored as an INI file called `negativelist.ini`. You can edit this list with a text editor if you need to add or remove negative files to suit your needs.

4. Press the **Scan Now** button () to scan the file.

   Setup Factory will analyze the file and determine any dependencies that it has. When it finds dependencies, it will attempt to determine the dependencies of those files as well. Setup Factory will then add all the dependency files to the list on the *Scan Dependencies* dialog.

5. *Optional:*
   Fine-tune the list of dependency files by removing any files that you don't want added to your Setup Factory project. Simply select any dependency files that you want to exclude, and press the **Remove** button () on the *Scan Dependencies* dialog.

6. Press the **OK** button to add the files on the *Scan Dependencies* dialog to your project.

NOTE

Dependency files are always added to the Archive tab.

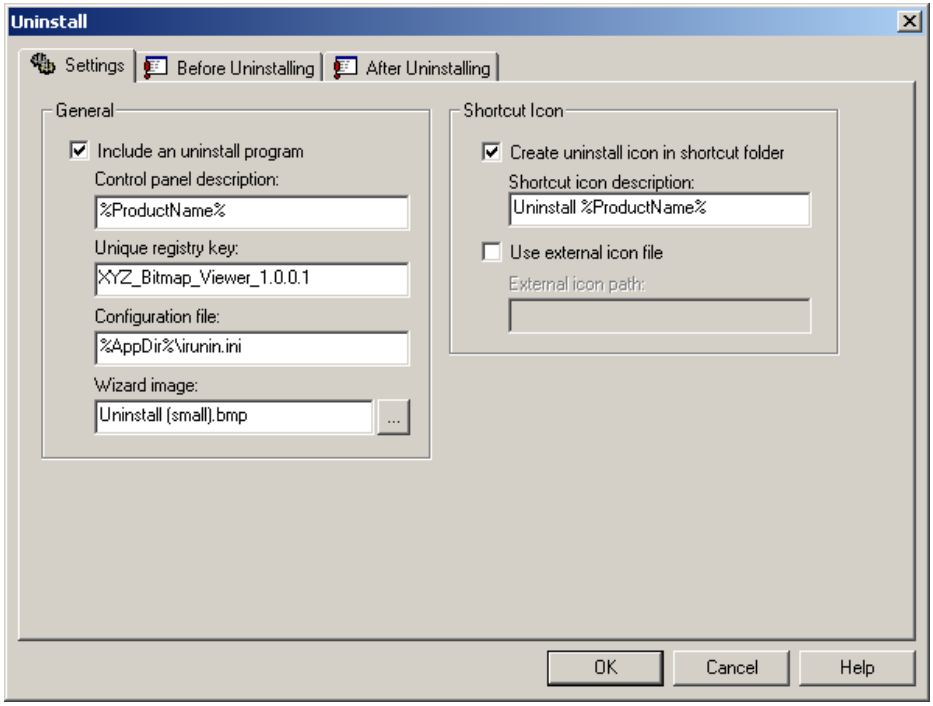
Chapter 13

Uninstall

Although as developers we have a hard time believing that someone would ever want to remove our software, it happens. After all, someone may need to remove a copy of your software from one system in order to install it on another without violating your license agreements. Fortunately, Setup Factory makes creating an uninstaller very easy.

The Uninstall Dialog

The *Uninstall* dialog is where you can configure the uninstallation settings for your project. You can access the *Uninstall* dialog by selecting **Design | Uninstall** from the menu, or by clicking on the Uninstall icon on the shortcut bar.



The Settings tab of the Uninstall dialog

There are three tabs on the *Uninstall* dialog: Settings, Before Uninstalling, and After Uninstalling.

The Settings Tab

The Settings tab is used to configure general options for the uninstaller, such as whether an uninstall item should be added to the Start menu, and what text should appear in the Add/Remove Programs component of the Windows control panel.

SEE ALSO

? For more information on the Settings tab, please consult the Command Reference.

The Before Uninstalling Tab

The Before Uninstalling tab is an action tab. This is where you can add any actions that you want performed *before* Setup Factory begins uninstalling the files you added to the Archive and CD-ROM tabs.

SEE ALSO

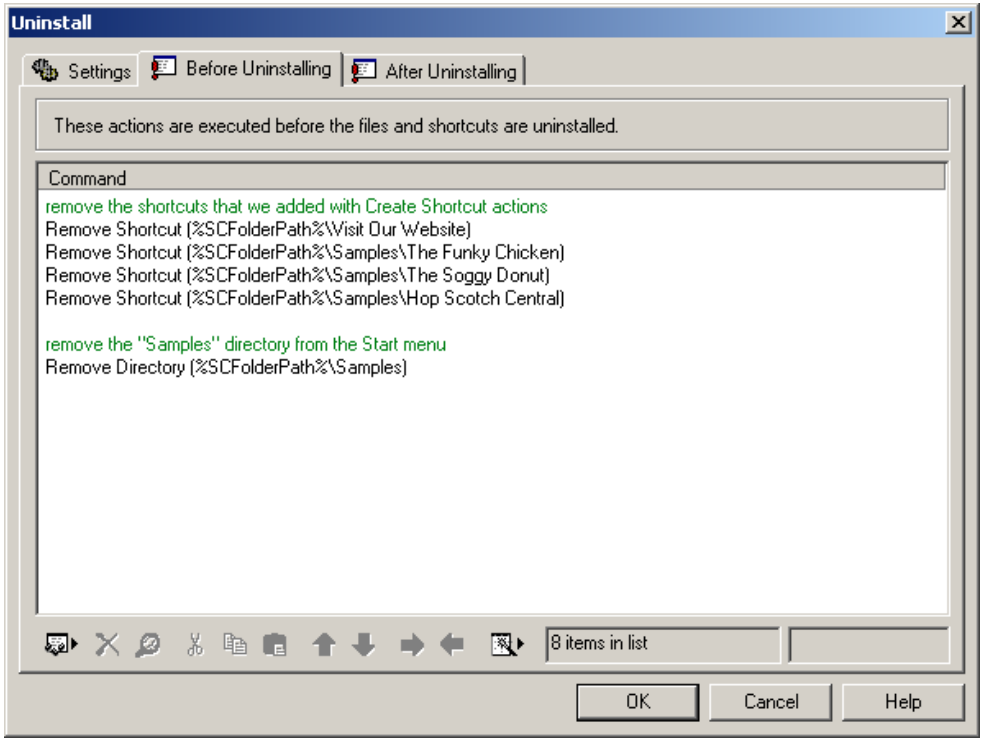
? For more information on the Before Uninstalling tab, please consult the Command Reference.

The After Uninstalling Tab

The After Uninstalling tab is an action tab. This is where you can add any actions that you want performed *after* Setup Factory finishes uninstalling the files you added to the Archive and CD-ROM tabs.

SEE ALSO

? For more information on the After Uninstalling tab, please consult the Command Reference.



The Before Uninstalling tab of the Uninstall dialog

NOTE

The action tabs on the *Uninstall* dialog allow you to "undo" any custom actions that were taken when your software was installed. You only need to use these tabs if your installer performs any actions during the installation that aren't automatically undone by the uninstaller.

The general rule of thumb is, if you add an action to do something during the installation process, and it needs to be undone when your software is uninstalled, you'll need to add actions to the *Uninstall* dialog to do so.

How the Uninstall Works

Before you add any uninstall actions, it's important to understand what the Setup Factory uninstaller automatically does for you.

By default, the uninstaller:

- Removes all of the files that were installed by the setup executable. In other words, any files that were listed on the project window at design time are automatically removed by default.

  There are, however, two exceptions to this rule:

  If you select the **Never remove** option for a file, that file will *not* be uninstalled automatically.

  If you select the **Shared/System file** option for a file, that file will only be uninstalled automatically if its usage count reaches zero during the uninstall. The usage count is the number of installed programs that require access to a specific registered file. This information is stored in the Registry for all shared files so they aren't removed if they're still required by another program.

  (Both of these options are configured on the Advanced tab of the *File Properties* dialog.)

- Removes any shortcuts and shortcut folders that were automatically created by the setup executable. This includes all of the shortcuts created using the Shortcuts tab of the *File Properties* dialog. It does *not* include any shortcuts created using Create Shortcut actions.

- Removes any folders automatically created by the setup executable, assuming they are empty after the automatically uninstalled files are uninstalled.

By default, the uninstaller does *not*:

- Remove any Registry entries created during the installation. You will need to remove your Registry entries manually by adding Modify Registry actions to the Before Uninstalling or After Uninstalling tab.

- Remove any shortcuts or shortcut folders created as the result of Create Shortcut actions. Only shortcuts and shortcut folders created using the Shortcuts tab of the *File Properties* dialog are removed automatically.

- Undo any INI file changes that were made during the installation. Since INI file changes are done with actions, you will need to use actions to undo them.

- Undo any text file changes that were made during the installation. Like INI files, text file modifications are done with actions and undone with actions.

- Remove files that were created by your software after installation.

- Remove files that were manually copied to your software's application directory by the user after installation.

- Remove files or directories that were copied, renamed, or moved during the installation. Of course, you can use the file and folder actions to make any changes you want with your uninstaller as well.

- Undo any changes made by external programs that were called from your installer using Execute Program actions.

- Undo any changes that were made to the user's system using actions. Anything you do with actions in the installer can only be undone with actions in the uninstaller.

Chapter 14

Variables

What Are Variables?

In Setup Factory, variables are special named "containers" for values that change. They're used to store information that won't be known until run time, like the location of the user's system directory, or the result of a Yes/No dialog presented to the user.

All variable names in Setup Factory begin and end with a percentage sign. At design time these variable names serve as placeholders, marking the places where the values will go once those values become known.

SEE ALSO

? For more information on variables, see page 35.

There are two kinds of variables in Setup Factory: built-in variables, and custom variables.

Built-in Variables

Setup Factory has a number of built-in variables that are already defined for you. They include:

- Directory and path variables such as %SrcDir% (the "source directory," where the Setup Factory installer is being run from), %WinDir% (the `Windows` directory), %SysDir% (the `Windows\System` directory), etc.

- System variables such as %OS% (the Operating System), %ScreenWidth% (the width of the user's display screen in pixels), %SysLanguage% (the numeric ID for the language being used on the user's system), etc.

- Product information variables such as %ProductName% (the name of your product), %CompanyName% (the name of your company), etc. (These variables are defined on the Product Info tab of the *General Design* dialog.)

- Time and date variables such as %JulianDate% (the number of days since midnight on January 1, 4713 B.C.), %CurrentMinute% (the current minute as set on the user's system), %CurrentYear% (the current year as set on the user's system), etc.

- Action-related variables such as %LastCommand% (the last action that was performed), %LastErrorNum% (a code indicating whether an error occurred while the last action was being performed), etc.

SEE ALSO

? For a complete list of the built-in variables, see *Built-in Variables* in the Command Reference, or see page 269 of this User's Guide.

Custom Variables

Custom variables are entirely up to you, and are specific to each project. Defining custom variables is easy—you just provide a variable name where a variable name is required. There are three places in Setup Factory where you can define custom variables:

Screen Properties dialogs

You can define variables to store the information received as a result of user input on many of the screens that you can include in your installer.

Action Properties dialogs

You can define variables using many of the actions in Setup Factory. These variables only come into effect as each action is performed.

Package Properties dialogs

You can define variables by setting up packages. Each package needs a unique custom variable that will be set to either true or false depending on whether the user selects or deselects that package.

What Can You Do With Variables?

Variables offer a lot of flexibility. You can use variables in the following ways:

1. You can use them to receive user input on screens with edit fields, list boxes, radio buttons, check boxes, and buttons.

2. You can use them to receive information from the Registry with a Read from Registry action, or to receive information from an INI file with a Read from INI File action.

3. You can use them in conditional expressions to set up run-time install conditions that determine whether individual files are installed.

4. You can use variables in the messages that get displayed on screens.

5. You can use them in screen conditions that determine whether individual screens are displayed.

6. You can use variables as temporary storage when manipulating strings with string actions.

7. You can use them in conditional expressions to set up conditional blocks and loops with IF and WHILE actions.

8. You can load the contents of a text file into a variable using a Read Text File action.

9. You can use actions to write their values to the Registry, to INI files or to text files.

10. You can use them in expressions to perform calculations on data retrieved from the user's system.

11. You can use them wherever you need to provide a path.

12. Basically, anywhere that you can enter text, you can use variables.

Defining Variables with Actions

Many of the actions in Setup Factory produce results that must be stored in variables. For example, the Read from Registry action reads a value from the Registry, and assigns that value to the variable of your choice. The value that is read from the Registry is the action's result, and the variable is where that result is stored.

Each of these actions has a field where you can provide the name of the variable you want the result to be stored in. If you provide the name of an existing variable, the result will overwrite that variable's contents. If you provide a new variable name, a new variable will be created automatically. The action's result will be stored in the variable you provide, regardless of whether the variable already exists.

Essentially, wherever a result needs to be stored, you can create a new variable "on the fly" by simply providing a variable name that isn't already being used in your project.

SEE ALSO

? For more information on the various actions, many of which allow you to define variables, please consult the Command Reference.

Defining Variables with Screens

Many of the screens in Setup Factory can receive input from the user. Whether that input is in the form of text entered in an edit field, or the checked or unchecked nature of a check box, it needs to be stored in a variable so it can be used by rest of the installer.

For example, the default *User Information* screen prompts the user to enter their name and their company's name into a pair of edit fields, and assigns the value of each field to the variable of your choice. The text that the user enters into each field is the value that will be assigned to the corresponding variable.

Another example: a variable is required for each check box on a *12 Check Boxes* screen. When the user selects a check box, the value "true" is assigned to the corresponding variable. If the user deselects that check box, the value "false" is assigned to the variable

instead. In other words, each check box has a variable that is set to either true or false to represent whether the check box was selected or not.

For every item on a screen that can receive user input, there is a field where you can provide the name of the variable you want the result to be stored in. If you provide the name of an existing variable, the result will overwrite that variable's contents. If you provide a new variable name, a new variable will be created automatically. The result of the user's input will be stored in the variable you provide, regardless of whether that variable already exists.

TIP

You can display the contents of a variable on a screen by including the variable name in the screen's message text.

A Little Common Sense Never Hurts

When defining and using variables, be sure to think about when and where they will be used, and what their values will be at that point in time. A variable that is only assigned a value at the end of the installation process won't have a value at the start.

For example, let's say you set up a *User Information* screen that asks the user for their name and stores it in a variable called %UserName%. It's important to realize that %UserName% won't contain the user's name until *after* that screen is shown. If the *User Information* screen is only displayed at the end of the installation process, %UserName% won't have a value at the beginning when your *Welcome* screen is shown. In this case, including %UserName% in the text of your *Welcome* screen would result in a message like "Hello %UserName%" instead of a message like "Hello Mary Lou."

The same is true for variables that you define on action tabs. When you assign a value to a variable using an action, the value doesn't get assigned to the variable until the moment the action is performed. A variable that you define on an action tab is only available to the actions that are listed below it on the tab, and to the actions on tabs that are performed "later" in the installation process. If you try to use the variable in another action higher "up" on the action tab, or on an action tab that happens earlier in the installation process, the variable won't have a value yet, and your installer probably won't work properly.

In other words, when you define a variable with an action, it is only available to the parts of the installation that are performed *after* that action. If you want to use a variable in a screen condition to control whether that screen is shown, make sure a value is assigned to the variable *before* it's that screen's turn to be displayed. If you want to use a variable in a run-time install condition, make sure a value is assigned to the variable before all the files are installed.

Naming Variables

You can name your custom variables anything you like, but there are a few guidelines to follow:

1. All variable names should begin and end with a percentage sign (%).

2. Variable names are not case sensitive—the names %MyVar% and %myvar% both refer to the same variable.

3. Don't use the same variable name twice unless you mean to.

   Using a name twice is fine if you're defining a variable in one place, and using it in another. Just be careful not to unintentionally give the same name to two different variables if they're being used to represent two different things.

   Of course, it's okay to define a variable twice if you *want* its value to change. In that case, the first value will only be in effect until the second one "overwrites" it.

4. Don't use a built-in variable name for one of your custom variables unless you know what you're doing.

   For example, calling a variable %SysDir% would override the built-in variable with the same name. Take time to familiarize yourself with the built-in variable names so you don't use any of them by accident.

5. Try to use meaningful names.

   For example, if you read a product's installation date from the Registry, naming the variable %Greegleborg452sx% probably isn't a good idea. Use a variable name like %InstallDate% instead.

6. Be consistent.

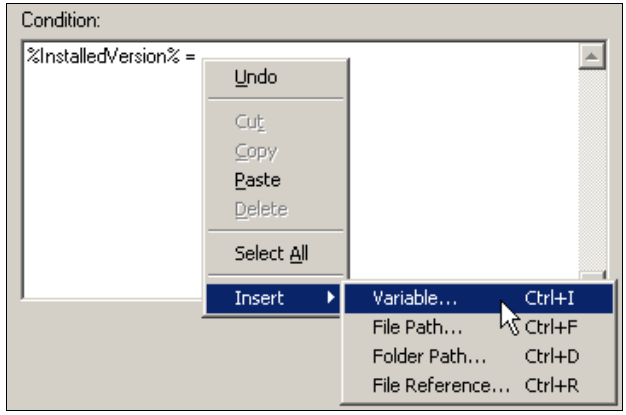
   If you start out with %UserName% and %UserAge%, don't name your third variable %WhereTheUserGoesToWork%. A name like %UserCompanyName% will be easier to remember when you find yourself asking "what did I name that variable, again?"

7. When in doubt, double-check.

   A common mistake is to name a variable one thing in one part of Setup Factory, and start calling it something else later. If you're not sure what name you used, double-check by going back to the screen where you defined the variable.

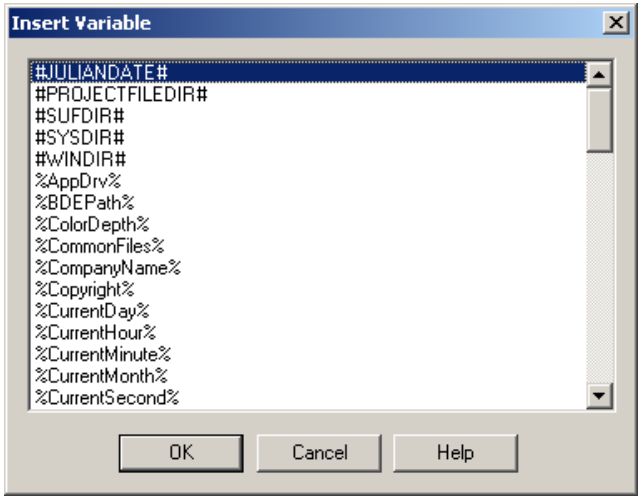
Inserting Variables

You can easily insert variables where applicable by right-clicking on an edit field and selecting **Insert | Variable** from the context menu.



The Insert | Variable item in the right-click context menu

This will open the *Insert Variable* dialog with all the built-in variables, custom variables, and design-time constants listed. Just select the variable you want to insert, and click OK.



The Insert Variable screen

TIP

Any custom variables you define in Setup Factory will automatically show up on the *Insert Variable* dialog.

Using Variables in Expressions

Before an expression is evaluated, any variable names in it are replaced by the values they represent. If these values contain spaces (or other value-delimiting characters) they can cause the expression to produce unexpected results, and even generate errors at run time.

SEE ALSO

? For more information on expressions, see page 221.

For instance, let's say we want to concatenate (i.e. join) the string " is a nice person" to the string contained in the variable %UserName%. We could use the expression:

%UserName% + " is a nice person"

...which works fine if the variable %UserName% contains a single word like *Mark*. But if %UserName% contains a space like, say, *Mark Smark*, the expression becomes:

Mark Smark + " is a nice person"

...which fails, because *Mark* and *Smark* are seen as two separate values with no operator between them. The space between *Mark* and *Smark* acts as a delimiter, "splitting" the words into two separate values. Setup Factory expects to see *Mark* <operator> *Smark*, but the operator is missing, and the resulting "broken" expression generates an error.

In order to avoid these problems, it's a good idea to surround variable names with quotation marks when using them in expressions. Any value between quotes is always interpreted as a single value, regardless of any spaces or other delimiting characters it might contain. By putting quotes around variables, each variable's contents are guaranteed to be seen as a single value.

So, if we rewrite our example expression as:

"%UserName%" + " is a nice person"

...we end up with:

"Mark Smark" + " is a nice person."

...which will not cause any problems.

NOTE

Variables only need to be quoted in two places: in expressions, and in the command line arguments for an execute command. You don't have to put quotes around variables anywhere else.

Value-delimiting characters

Delimiting characters separate values in expressions. In Setup Factory, spaces and all the operators except AND, OR and MOD act as delimiters in unquoted strings. These characters will delimit or "break" a string into multiple parts unless the string is surrounded by quotation marks.

In order to be seen as a single value, a variable must be surrounded by quotation marks if it contains spaces or any of the following characters:

+ - * / = > < ( )

Examples:	"2+2"	(single value, not delimited)
	2-2	(same as: "2" - "2")
	score	(single value, not delimited)
	yes/no	(same as: "yes" / "no")

Chapter 15

Expressions

What Are Expressions?

An expression is any valid combination of *values* and *operators* that resolves to a single result. For instance, in the expression 1 + 2, "1" and "2" are values, "+" is the add operator, and the resulting value would be 3.

You can use expressions to perform calculations, compare values, set conditions and make decisions at run time.

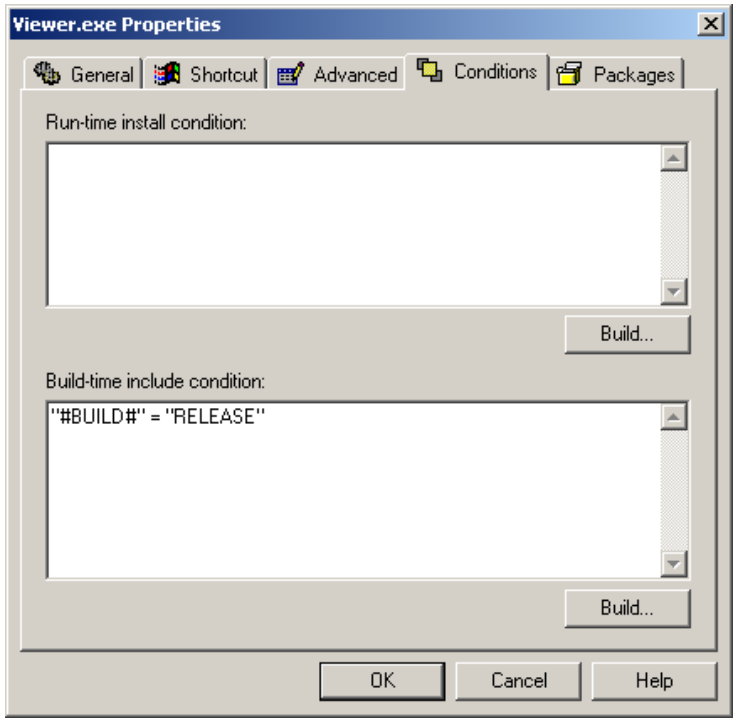
Where Can You Use Them?

There are five places where expressions can be used in Setup Factory: in build-time conditions, in run-time conditions, in screen conditions, in conditions for IF and WHILE actions, and in Assign Value actions.

Build-time Conditions

You can use expressions to set up build-time conditions. Build-time conditions determine whether individual files are included in the installer.

Each file in a Setup Factory project has a build-time condition associated with it. You can edit the build-time condition in the **Build-time include condition** field, which is found on the Conditions tab of the *File Properties* dialog.



A build-time condition on the Conditions tab

Before any file on the Archive tab is added to the setup executable, its build-time condition is evaluated, and the result is interpreted as a Boolean value. If the result is true, the build-time condition has been met, and the file will be included in the setup executable. If the result is false, the condition has not been met, and the file won't be included in the setup executable.

For a file on the CD-ROM tab, the build-time condition determines whether the installer will be aware of the file. Files on the CD-ROM tab are never included in the setup executable, but their build-time conditions can affect whether the installer creates any shortcuts for them, or includes them in any disk space calculations that are performed.

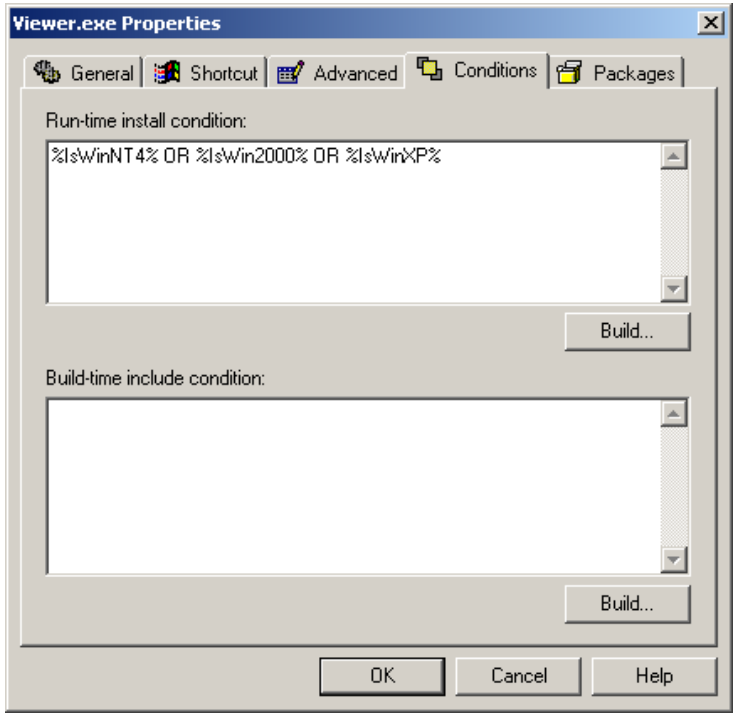
NOTE

If the build-time condition for a file is not met, the installer will operate as though that file was never added to the project at all.

Run-time Conditions

You can use expressions to set up run-time conditions. Run-time conditions determine whether individual files are installed on the user's system.

Each file in a Setup Factory project has a run-time condition associated with it. You can edit the run-time condition in the **Run-time install condition** field, which is found on the Condition tab of the *File Properties* dialog.



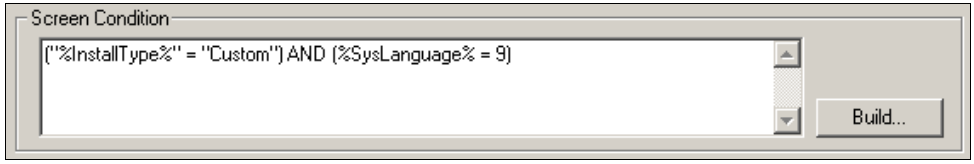
A run-time condition on the Conditions tab

Before a file is installed, its run-time condition is evaluated, and the result is interpreted as a Boolean value. If the result is true, the run-time condition has been met, and the file is installed on the user's system. (Assuming, of course, that any other conditions for the file are met, such as the file's overwrite settings, and—if the file belongs to a package—whether the appropriate package was selected.) If the result is false, the condition has not been met, and the file won't be installed.

Screen Conditions

You can use expressions to set up screen conditions. Screen conditions determine whether individual screens in the installer are displayed at run time.

Each screen in a Setup Factory project has a screen condition associated with it. You can edit the screen condition in the **Screen Condition** field, which is found on the Settings tab of the *Screen Properties* dialog.



The default screen condition for the Select Packages screen

Before a screen is displayed, its screen condition is evaluated, and the result is interpreted as a Boolean value. If the result is true, the screen condition has been met, and the screen will be displayed. If the result is false, the condition has not been met, and the screen will not be displayed.

NOTE

Screen conditions are evaluated before any of the actions on the screen's Before tab are performed. If the screen conditions aren't met, none of the actions attached to that screen are performed.

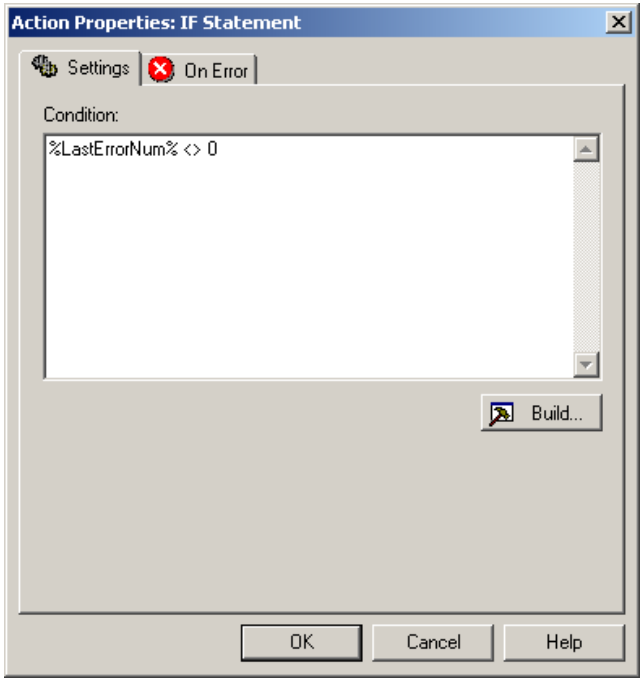
IF and WHILE actions

You can use expressions to set up IF and WHILE actions. The IF action causes every action between it and the corresponding END IF action to be performed only if its expression evaluates to a true result. The WHILE action continues performing every action between it and the corresponding END WHILE action until its expression evaluates to a false result.

```
IF (%UserName% = "Joe Blow")
    These actions are only performed if the variable...
    ...%UserName% contains the string "Joe Blow"
    Show Message Box (Hello Joe!)
END IF
```

Example of IF and END IF actions in use

You can enter expressions in the **Condition** field on any *Action Properties: IF Statement* or *Action Properties: WHILE* dialog.

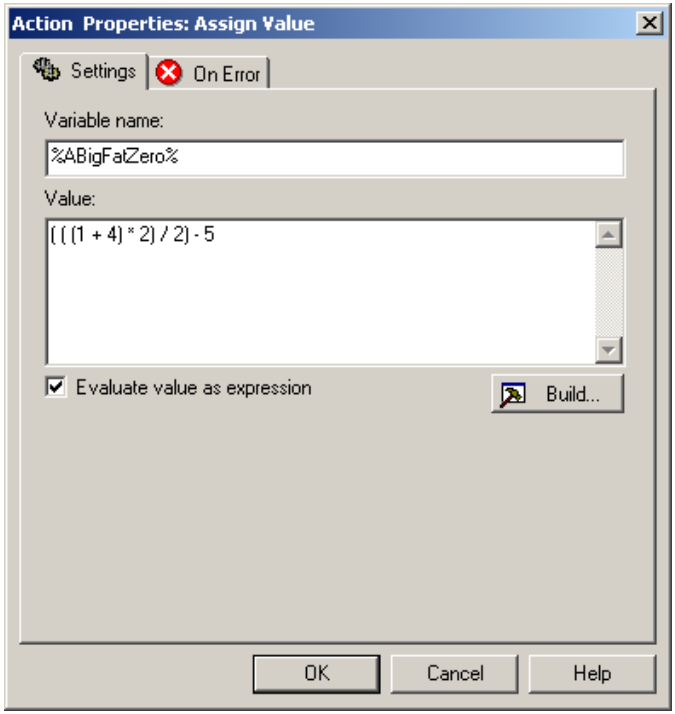


An expression in the Condition field for an IF action

Assign Value actions

You can use expressions to calculate values on *Action Properties: Assign Value* dialogs. These dialogs have a special **Evaluate value as expression** check box that determines how the text in the **Value** field is interpreted.

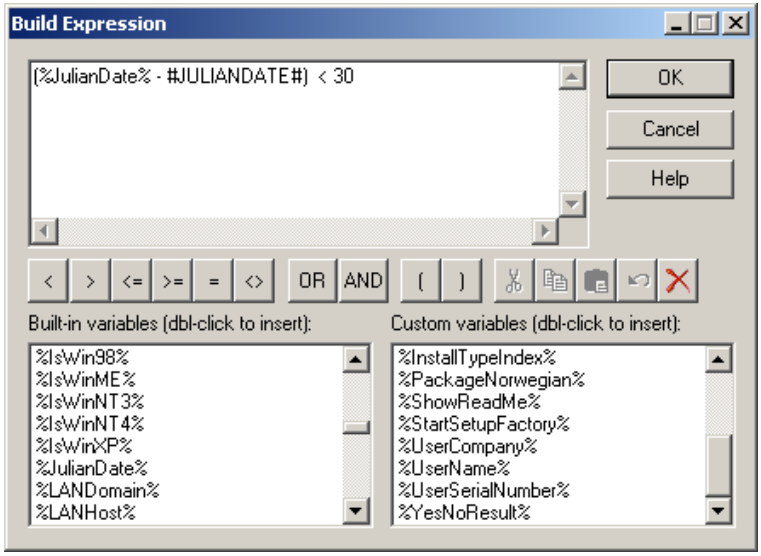
When the **Evaluate value as expression** check box is selected, anything you type into the **Value** field is evaluated at run time, and the result of the evaluated expression is then assigned to the variable. When the check box is not selected, whatever you type in the **Value** field is assigned to the variable without being evaluated first.



An expression in the Value field for an Assign Value action

TIP

On any dialog where you can enter an expression, you can press the **Build Expression** button () to open the *Build Expression* dialog.



The Build Expression screen

Values

There are five kinds of values that you can use in Setup Factory expressions: integers, real numbers, strings, versions and variables.

Integers

Integers consist of whole numbers, like 1, -32516 and 475. They can be written with or without quotation marks, e.g. "341" and 341 are equivalent.

Real Numbers

Real numbers consist of "fractional" or "floating point" values, like 0.594, -1.5 and 123.654. They can be written with our without quotation marks—for example, "0.5" and 0.5 are equivalent. Any leading zero can be omitted as well, so, for example, ".75" and "0.75" are equivalent.

Strings

Strings consist of any sequence of characters surrounded by quotation marks, like "Wilbur", "the 25 happiest days of summer" and "hello world".

Versions

Versions are special strings that begin with the letter "v" followed by numbers which can be separated by one or more periods. (If there are two or more periods, the letter "v" can be omitted.) Examples: "v2", "v4.0", "v1.03", "0.2.0.0", and "6.0.1".

Variables

You can use variable names as placeholders for integers, strings or versions in an expression. Before an expression is evaluated, any variable names in the expression are converted to the values they represent.

IMPORTANT

! You should always surround variable names with quotation marks whenever they're used as values in expressions, in case they're assigned values that contain spaces. This way, a variable like "%fullname%" becomes a single string like "Joe Blow" instead of being interpreted as two string values like "Joe" and "Blow", which would generate an error.

SEE ALSO

? For more information on using variables in expressions, see page 219.

Expressions can also be used as "values" in a more complex expression. For instance, in the expression (1 + 2) – 3, the sub-expression (1 + 2) is evaluated first, resulting in the value 3. Then the expression 3 – 3 is evaluated, resulting in the final value 0.

Boolean Values (True and False)

Boolean values are used to describe logical truths—whether something is "true" or "false."

In Setup Factory 6.0, an expression is considered *true* if it resolves to either the word "true" or any non-zero integer value. An expression is considered *false* if it resolves to the number 0 or any string other than "true".

For example, an expression that resolves to "-26" is considered true. An expression that resolves to "0" is considered false. An expression that resolves to "True" is true, and an expression that resolves to "yikes!" is false.

Real numbers are a bit of a special case: a real number only resolves to true if it is greater than or equal to 1.0, or less than or equal to -1.0. In other words, if a real number falls between -1.0 and 1.0, it's false.

This is because the fractional part of a real number is *discarded* before the number is interpreted as a true or false value. So, "1.5" becomes "1", which resolves to *true*. "0.0" and "0.9" both become "0", so they both resolve to *false*.

Another way of looking at this is, if the part of the number to the left of the decimal is zero, the real number resolves to false...otherwise, it resolves to true.

So, for example, an expression that resolves to "-27.3" is considered true, but an expression that resolves to "-0.5" is considered false. An expression that resolves to "1.0" is true, and an expression that resolves to "0.99999999999" is false.

How values are stored
All values are stored internally as strings. It's the content of the strings that determines how they're interpreted at run time.

A string containing only numeric characters is considered an integer when it's used in an expression. A string consisting of numbers and a single decimal point is considered a real number. A string consisting of numbers separated by 2 or more decimal points is considered a version. A string that begins with the letter "v" followed by numbers (that may or may not be separated by decimal points) is also considered a version. Anything else is assumed to just be a string.

Operators

Operators are used in expressions to perform specific actions on one or more *operands*, generating a single return value or result. Operands are simply the values or expressions that each operator operates on. For instance, in the expression 1 + 2, the values "1" and "2" are the operands, and "+" is the symbol for the add operator.

Operators are said to have *precedence*, which is a way of describing the rules that determine which operations in a series of sub-expressions get performed first. A simple example would be the expression 1 + 2 * 3. In Setup Factory, the multiply (*) operator has *higher precedence* than the add (+) operator, so this expression is equivalent to 1 + (2 * 3). In other words, the sub-expression 2 * 3 is performed first, and then 1 + 6 is performed, resulting in the final value 7.

You can override the natural order of precedence by using parentheses. For instance, the expression (1 + 2) * 3 resolves to 9. The parentheses make the whole sub-expression 1 + 2 the left operand of the multiply (*) operator. Essentially, the sub-expression 1 + 2 is evaluated first, and the result is then used in the expression 3 * 3.

Operators are also said to have *associativity*, which is a way of describing which sub-expressions are performed first when the operators have equal precedence. In Setup Factory, most operators are *left associative*, which means that whenever two operators have the same precedence, the operation on the left is performed first. (The only exceptions are the unary plus, the unary minus, and the logical NOT operators, all three of which are right-associative.)

The left-associativity of the subtract (-) operator is why the expression 10 - 5 - 2 resolves to 3 instead of 7. It's interpreted as (10 - 5) - 2, and not 10 - (5 - 2).

Table of Operator Precedence and Associativity

The following operators can be used in Setup Factory expressions.

Notes: A **unary** operator takes a single value.
A **binary** operator takes two values.
A **right-associative** operator operates on the value to its right.
An **infix** operator operates on one value to its left and one value to its right.
(All infix operators in Setup Factory are left-associative.)

In order of precedence, from highest to lowest:

	Name:	**Notes:**	**Precedence level:**
(	open parenthesis		*8 (highest)*
)	closed parenthesis		*8*
+	unary plus	unary, right-associative	*7*
-	unary minus	unary, right-associative	*7*
!	logical not	unary, right-associative	*7*
*****	multiply	binary, infix	*6*
/	divide	binary, infix	*6*
MOD	modulus	binary, infix	*6*
+	add	binary, infix	*5*
-	subtract	binary, infix	*5*
<	less than	binary, infix	*4*
<=	less than or equal	binary, infix	*4*
>	greater than	binary, infix	*4*
>=	greater than or equal	binary, infix	*4*
=	equal	binary, infix	*3*
!= or **<>**	not equal	binary, infix	*3*
AND	Boolean AND	binary, infix	*2*
OR	Boolean OR	binary, infix	*1 (lowest)*

Parentheses

Parentheses are used to group sub-expressions and override the rules for precedence and associativity. Anything between an open parenthesis and a closed parenthesis is resolved first, before the rest of the expression is evaluated.

For instance, in the expression (5 + 2) * 3, the part between parentheses is performed first, and the result (7) is then used as a value in the larger expression 7 * 3. If the parentheses were omitted, the expression 5 + 2 * 3 would resolve to 11 instead.

You can "nest" parentheses to form more complex expressions. Nesting just means using parenthetical expressions inside other parenthetical expressions. For example, the expression -((2 + 4) * 2) has a nested parenthetical expression and resolves to -12.

TIP

Use parentheses whenever you can to help make your expressions easier to read. 10 + (2 * 5) requires less thought to interpret than 10 + 2 * 5.

Logical (Boolean) Operators

Logical operators are used to combine the results of Boolean expressions. A Boolean expression is just like any other expression, but its result is evaluated to either true (any non-zero value) or false (0). In Setup Factory, true and false are represented by 1 and 0.

There are three logical operators in Setup Factory: AND, OR and !.

AND	And	Returns 1 (true) if both of its values are true. Returns 0 (false) otherwise. *Example: 23 AND 0 resolves to 0*
OR	Or	Returns 1 (true) if either of its values are true. Returns 0 (false) if both of its values are false. *Example: 5 OR 0 resolves to 1*
!	Not	Returns the Boolean opposite of its value, which is 1 (true) if its value is false, and 0 (false) if its value is true. *Example: !5 resolves to 0*

Relational Operators

Relational operators are used to compare two values. Relational expressions resolve to a Boolean (true/false) value: a true result resolves to 1, and a false result resolves to 0. For example, 23 > 4 is true because 23 is greater than 4, so this expression resolves to 1. "orange" < "apple" is false because the string "orange" is not alphabetically lower than "apple", so this expression resolves to 0.

When versions are compared, each individual number in one version is compared with the corresponding number from the other. So, "1.3.4" < "1.10.4" is true, because the "3" in the middle of the first version is less than the "10" in the middle of the other.

There are seven relational operators in Setup Factory: =, >, >=, <, <=, <> and !=.

=	Equal	Returns 1 (true) if the value on its left is equal to the value on its right. Returns 0 (false) if its two values are not the same. *Example: 4 = 4 resolves to 1*
>	Greater than	Returns 1 (true) if the value on its left is greater than the value on its right. Returns 0 (false) otherwise. *Example: 5 > 20 resolves to 0*
>=	Greater than or equal	Returns 1 (true) if the value on its left is greater than or equal to the value on its right. Returns 0 (false) otherwise. *Example: 5 >= 5 resolves to 1*
<	Less than	Returns 1 (true) if the value on its left is less than the value on its right. Returns 0 (false) otherwise. *Example: 5 < 20 resolves to 1*
<=	Less than or equal	Returns 1 (true) if the value on its left is less than or equal to the value on its right. Returns 0 (false) otherwise. *Example: "bart" <= "lisa" resolves to 1*
<>	Not equal	Returns 1 (true) if the value on its left is not equal to the value on its right. Returns 0 (false) if both values are the same. *Example: "1.4.7" <> "1.4.8" resolves to 1*
!=	Not equal	Same as the <> operator above. *Example: 3 != 3 resolves to 0*

Arithmetic Operators

There are seven arithmetic operators in Setup Factory: plus, minus, +, -, * , / and MOD.

+	Unary plus	Indicates a positive value. *Example: +7 resolves to 7*
-	Unary minus	Forms a negative value. *Example: -4 resolves to -4*
+	Add	Adds two values together. *Example: 2 + 3 resolves to 5*
-	Subtract	Subtracts one value from another. *Example: 150 - 120 resolves to 30*
*****	Multiply	Multiplies one value by another. *Example: 5 * 3 resolves to 15*
/	Divide	Divides one value by another. If both values are integers, integer division is performed—any remainder is discarded. *Example: 21 / 2 resolves to 10*
MOD	Modulus	Returns the remainder after a division is performed. *Example: 21 MOD 2 resolves to 1*

String Operators

There are three special string operators in Setup Factory: +, - and *.

+	Add (concatenate)	Concatenates two strings, i.e. appends one string to another. *Example: "hello" + "world" resolves to helloworld*
-	Subtract (strip)	Strips one string from another, i.e. removes every occurrence of the string on the right from the string on the left. *Example: "happy apple" - "p" resolves to hay ale*
*****	Multiply (repeat)	Repeats a string a given number of times. *Example: "Apple" * 3 resolves to AppleAppleApple*

Version Operators

There are two special version operators in Setup Factory: + and -.

+	Add	Adds two versions together. *Example: "1.3.2" + "v1.7" resolves to 2.10.2*
-	Subtract	Subtracts each element of one version from the corresponding element of another. Negative results are "truncated" to zero. *Example: "2.0.5" - "v1.9" resolves to 1.0.5*

NOTE

The seven relational operators (=, >, >=, <, <=, <> and !=) can also be used with versions. When versions are compared, each individual number in one version is compared with the corresponding number from the other.

Notes

1. Anything between quotes (""), including whitespace, is considered a single value.

2. An expression is considered *true* if it resolves to either the word "true" or any non-zero integer value. An expression is considered *false* if it resolves to anything else.

3. A real number is considered true if it is greater than or equal to 1.0, or less than or equal to -1.0. In other words, if a real number falls between -1.0 and 1.0, it's false. For example, 1.0 is considered true, and 0.99999 is considered false.

4. The strings "true" and "false" can be used to represent the two Boolean values in Setup Factory expressions. For example, the expression *true or false* is equivalent to the expression *1 or 0*. Both expressions resolve to 1.

5. "True" and "false" are not case sensitive. "True", "true", "tRuE" and "TRUE" all resolve to 1. "False", "false", "fAlsE" and "FALSE" all resolve to 0.

6. Non-quoted strings are delimited by whitespace and any of the operators *except* MOD, AND and OR. For instance, sandy is seen as the single value "sandy" and not the expression "s and y".

7. You can perform string concatenation with the add operator ('+').

8. You can compare strings alphabetically using the <, <=, >, >=, =, != and <> operators.

9. String comparisons are case insensitive. "Hello world" is equal to "hEllO WoRlD" in Setup Factory.

10. You can "multiply" strings with an integer value. (2 * "Ha" resolves to HaHa)

11. You can "strip" every occurrence of one string from another with the subtract operator ('-').

12. All stand-alone strings except "true" resolve to 0 (false) when a Boolean result is expected. For instance, the expression "hello" resolves to 0 (false) when used alone in an IF statement.

13. With the exception of "true", strings resolve to 0 (false) when used with the Boolean infix operators AND and OR. For instance, *1 AND "hello"* is equivalent to *1 and 0* and resolves to 0 (false).

14. Strings that begin with the letter "v" followed by only numbers and periods are interpreted as versions. A string consisting of only numbers separated by two or more periods is automatically considered a version even if the letter "v" is omitted.

15. The "v" prefix is omitted from the result when a version calculation is performed. For example, v1.0 + v2.3 resolves to 3.3, and not v3.3.

16. You can compare versions using <, <=, >, >=, =, != and <>, and you can perform "corresponding element addition" and "corresponding element subtraction" on them using + and -. "Version.revision" notation is assumed, so v1.1 + v1.9 is 2.10 ("the 10^{th} revision of version 2"), not 3.0. After a calculation, any negative numbers in a version string are reduced to 0. For example, 2.3.5 - 7.1.6 resolves to 0.2.0.

Syntax Rules

The following rules describe the various syntax checks that are performed on all expressions in Setup Factory:

1. Each '(' must have a matching ')' and vice-versa.

   Good: ((%a%))

   Bad: ((%a%)

2. No empty parentheses. '( )' is not allowed.

   Good: (%BelovedGazeInThineOwnHeart%)

   Bad: ()

3. No open parenthesis immediately after a closed parenthesis. ')(' isn't allowed.

   Good: (%foo%) > (%bar%)
   (2 > 3) OR ("tree" <= "tree-house")

   Bad: (%foo%)(%bar%)
   (2 > 3)("tree" <= "tree-house")

4. Each closed parenthesis must follow a value or a closed parenthesis. Only '<value>)' and '))' are allowed.

   Good: (%foo% = (%bar% + "hello"))
   ((1 < 2) AND (2 < 3))

   Bad: (%foo% =)
   3 * (5 +)

5. Only open parentheses and right-associative operators are allowed before the first value in an expression. An expression can't begin with a binary infix operator or a closed parentheses.

   Good: !7
   !(!(!0))

   Bad: * %foo%
   !(/ 4)

6. You can't have two values in a row without an operator between them. '<value><value>' isn't allowed.

   Good: %foo% = %bar%

   Bad: %foo% %bar%
   "apple" "jack"

7. You can't have a value immediately before an open parenthesis. '<value>(' isn't allowed.

   Good: "hello" + ("wo" + "rld")

   Bad: "hello" ("wo" + "rld")

8. You can't have a value immediately after a closed parenthesis. ')<value>' isn't allowed.

   Good: (2 + 4) + "th day violation"
   (%count% + 1) > 5

   Bad: (2 + 4) "th day violation"
   (%count% + 1) 5

9. All right-associative operators must be followed by a value, with no other operators between the right-associative operator and its value *except* for open parentheses and other NOT ('!') operators. In other words, the NOT operator is allowed to repeat, but not the unary plus or minus...you can have '!!!<value>', but not '++++<value>'.

   Good:
   ```
   !((-(-1)))
   !!!!!1
   +( -( -2 ) )
   ```

   Bad:
   ```
   !+-%foo%
   !!-6
   -!2
   --(2)
   +-5
   ```

10. Every infix operator must have a value to the left of it.

    Good:
    ```
    %foo% + %bar%
    4 * 5 AND 2 + 3
    ```

    Bad:
    ```
    / 12
    AND 26
    * 5 + 2
    ```

11. The last token in an expression must be a value or a closed parenthesis. It can't be an infix operator, right-associative operator or open parenthesis.

    Good:
    ```
    2 + 3
    (5 > 2)
    ```

    Bad:
    ```
    2 +
    5 >
    ```

Chapter 16

Supporting Multiple Languages

One of Setup Factory's strongest features is its support for creating foreign language installations. You can use Setup Factory to create an installer that will automatically display messages and prompts in your user's native language.

TIP

The multi-language feature also allows you to customize the English text displayed by "built-in" messages and prompts at run time.

There are four steps to creating a multilingual installer: translating screens, translating language files, translating packages, and translating actions.

Translating Screens

The first step in creating a multilingual installer is to translate all of your screens into the languages that you want to support. This can be done from the *Screens* dialog. To access the *Screens* dialog, select **Design | Screens** from the menu, or click on the Screens icon in the shortcut bar.

Create one screen of each type (*Welcome* screen, *Select Install Folder* screen, etc.) for each language that you want to support. It's a good idea to include the name of the language in the name that you assign to each screen—names like "Welcome (French)" make it easy to identify which language a screen has been translated to.

Edit the screen properties to translate any text that is displayed on the screen, including text that appears on the title bar and navigation buttons.

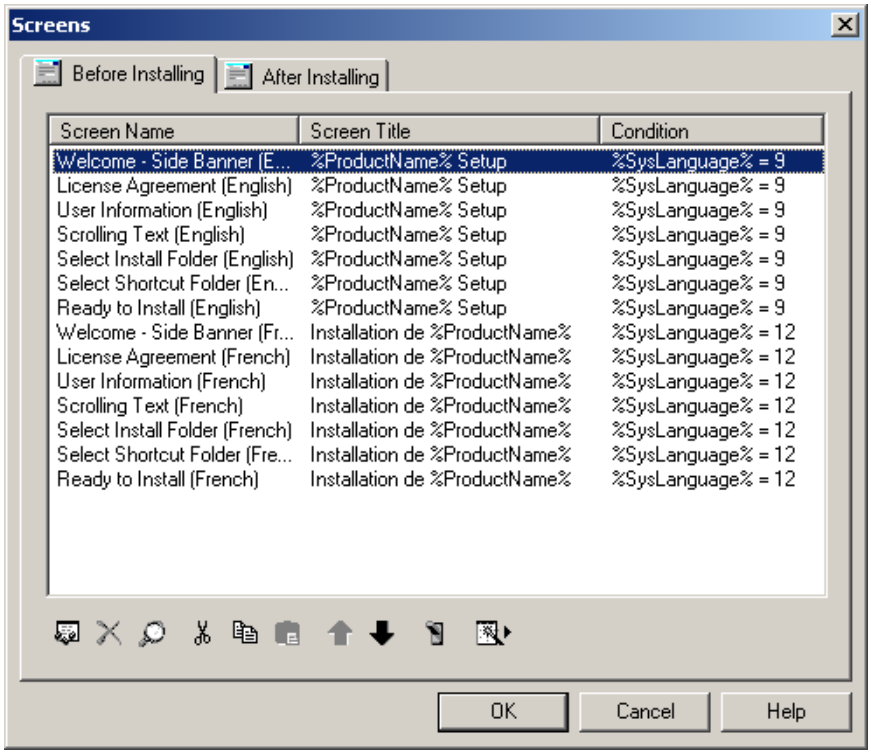
Then, set up a screen condition so the screen will only be displayed if the appropriate language is being used on the user's system. You can use the built-in variable

%SysLanguage% to compare the language ID for the user's system to the language ID for the screen's language. For example, a condition like %SysLanguage% = 10 would cause a screen to only be displayed if the user's system language was Spanish.

Once you've finished translating a screen, you might want to export it using the **Advanced operations** button ( ). This way you can add the translated screen to future projects without having to translate the original screen again.

SEE ALSO

? For more information on the *Screens* dialog, see *Screens* in the Command Reference, or see page 142 in this User's Guide.



Translated screens on the Screens dialog

Translating Language Files

Once your screens are translated, the next thing that requires translation are the other messages, buttons and prompts that can appear during an installation. These "built-in" messages are stored in Setup Factory language (`.lng`) files.

Language files are special INI files used by Setup Factory. Each language file corresponds to a Windows language ID, and includes all of the "built-in" messages for that language in plain text format.

NOTE

Setup Factory comes prepared with language files for many common languages. You can find the Setup Factory language files in your `C:\Program Files\Setup Factory\Languages` folder.

To translate a language file:

1. Make a copy of the language file you want to translate and name it according to the new language, e.g. copy "`English.lng`" to "`German.lng`".

2. Open the new language file in a text editor and change the language ID to the appropriate ID for the new language. For example, if you were translating the English messages to German, you would change the line that reads:

   ```
   LanguageID=9
   ```

   to:

   ```
   LanguageID=7
   ```

   A complete list of language IDs can be found in the `C:\Program Files\Setup Factory\languages\langids.ini` file.

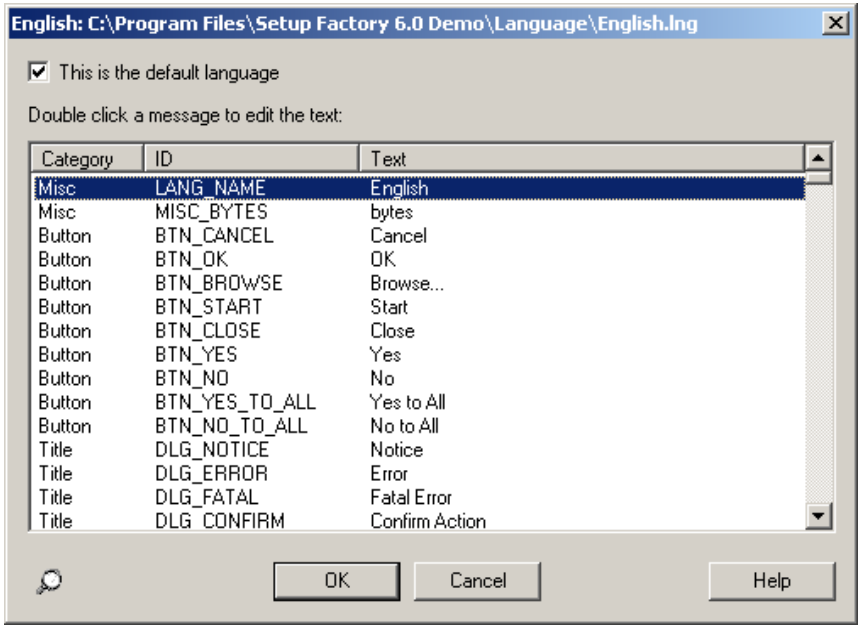
3. Add the new language file to your Setup Factory project on the Languages tab of the *General Design* dialog.

   You can access the *General Design* dialog by selecting **Design | General Design** from the menu, or by clicking on the General Design icon on the shortcut bar.

4. Select the new language in the list, and press the **Properties** button ( ) to translate the messages using the *Language Module Messages* dialog.

TIP

You can also edit the messages using your text editor, and then add the translated language file to your Setup Factory project.



The Language Module Messages dialog

Translating Packages

The next step in creating a multilingual installer is to translate the names and descriptions of your packages. Of course, this step is only necessary if you are using packages in your installer.

To translate your packages:

1. Select **Design | Packages** from the menu.
2. Select the package that you want to translate.
3. Press the **Edit Package** button (). This will open the *Package Properties* dialog.
4. Switch to the Localize tab.
5. Press the **Add** button (). This will open the *Localize Package* dialog.
6. Select the language that you wish to translate the package to in the **Language** drop-down list.
7. Enter the translated text for the name and description of the package in the **Name** and **Description** fields.
8. Press the **OK** button to accept the translation.

Translating Actions

The final (and completely optional) step in creating a multilingual installer is to translate any custom messages displayed as the result of actions.

There are two ways that actions can cause messages to be displayed: a message can be displayed using a Show Message Box or Yes/No Message Box action, and a custom error message can be displayed when an error occurs.

The best way to handle both of these cases is to do what programmers call "adding indirection." Instead of entering the message text directly into the action or custom error message, define a custom variable to hold the message instead. Then use a series of IF actions to assign the appropriate translation to the variable, depending on what language ID is detected on the user's system.

For example, to display a message in either English, French, or German, you could use actions that look something like this:

```
English message
IF (%SysLanguage% = 9)
    Assign Value (%message% = my message in english)
END IF

French message
IF (%SysLanguage% = 12)
    Assign Value (%message% = mon message en français)
END IF

German message
IF (%SysLanguage% = 7)
    Assign Value (%message% = meine meldung auf deutsch)
END IF

Show Message Box (%message%)
```

SEE ALSO

? For more information on supporting multiple languages, please consult the Command Reference.

Chapter 17

Creating a CD-ROM Installer

In the past few years, CD-ROM has become the distribution medium of choice. Setup Factory is perfectly suited to meet the needs of a CD-ROM installation. Most CD-ROM installs will fall into one of the following three scenarios:

- You want to install all of the files to the user's hard drive. In this case you are just using the CD-ROM as a means of distributing your setup, and once the software is installed to the hard drive, the CD-ROM is no longer used to operate the program.

- You want your files to remain on the CD-ROM so the user can run the program from there. In this scenario, you'll need to install some shortcuts on the user's system, and possibly make some system changes to support your software. The shortcuts will allow the user to easily run the program files directly off the CD-ROM.

- You have a large distribution that users with lots of hard drive space may want to install for performance reasons, but that others will want to run from the CD. In this case, you want to let the user decide which option is best for them. You could even let them choose which files they want to install, and which files they want to leave on the CD-ROM.

Each of these scenarios is examined in detail below.

Full Install to the Hard Drive

In this scenario, you want to install all of your files to the user's hard drive, and the CD-ROM is just a distribution medium for the installer, just like floppy disks or an Internet download would be. The only difference is that you need to decide whether to compress the files into the setup executable, or leave them uncompressed on the CD-ROM and install them from there. (You can ship files separately when you're using other distribution methods, but it's uniquely convenient to do so when you're distributing software on CDs.)

Here are some advantages to including the files in the setup archive:

- The files will be compressed and will require less space on the CD-ROM. This could be important if your CD is packed with data and you need every bit of space to count.

- The setup executable is self-contained and portable. The same executable that you distribute on CD-ROM can easily be placed on your web site for download.

- The setup archive is secure. The files in the setup executable are compressed and cannot be read from outside the setup. This way, you can use a *Verify Serial Number* screen to protect the data in your installer, and not have to worry about unauthorized access to your files on the CD-ROM.

Here are some advantages to excluding the files from the setup archive and just having them remain uncompressed on the CD-ROM:

- The files will be accessible from the CD-ROM, so your users can access the files any way they want to.

- You'll be able to update the files independently. If you make minor changes to one or more files, you won't have to rebuild the setup executable. You can just replace the old files with new versions on your master CD.

NOTE

Contrary to what you might expect, there is really no performance difference between installing files from the setup executable and installing them directly from the CD-ROM. Any performance loss from having to decompress large files in the setup archive is made up by not having to read as much data off the CD.

Also, Setup Factory's advanced file streaming engine doesn't require any more temporary space on the user's hard drive when it installs files from the setup executable than it does when it installs files directly off the CD-ROM.

Of course, you aren't limited to just one or the other; you can distribute some of the files in your project in the setup executable, and some of the files "out bare" on the CD-ROM. Once you've decided how you want to distribute the files, just add the files that you want in the setup to the Archive tab, and add the files that you want to leave on the CD to the CD-ROM tab. The rest of the installation process is the same as it is for any other installation.

Leaving All Files on the CD-ROM

In this scenario, you always want your program to be run from and remain on the CD-ROM. For example, you may be distributing large multimedia applications or extensive online help systems that your users have said they prefer not to install.

In this case, you don't need to include any files in your Setup Factory project at all. You'll probably just need to create shortcuts on the Start menu to give your users easy access to the application on your CD-ROM. Of course, if your application requires any changes made to the user's system, such as Registry entries to personalize the software, the installer will also need to make those changes.

To create shortcuts to files on the CD-ROM, you can either use the Shortcut tab of the *File Properties* dialog, or you can use Create Shortcut actions.

When using Create Shortcut actions, you would enter the path to the file on the CD-ROM in the **Target file** field. Since you probably don't know what drive letter will be assigned to the user's CD-ROM drive, you'll need to use the built-in variable %SrcDrv% in this path. %SrcDrv% represents the letter of the drive where the setup executable is being run from. Assuming that your setup executable is run from the CD-ROM, you can use %SrcDrv% to represent the user's CD-ROM drive letter in your paths.

So, for example, if you wanted to make a shortcut to a file called `MyApp.exe` on the root of your CD-ROM, you would set the **Target file** field to `%SrcDrv%\MyApp.exe`.

Letting the User Choose

In this scenario, you want to give your users the choice of whether to install the application to their hard drive or run it from the CD-ROM. The main difference between this scenario and the previous ones is that you will use the *Packages* dialog to create packages for your files so the user can choose what type of installation they want.

To create this type of installer:

1. Add all of your files to the CD-ROM tab. This way you aren't storing two copies of every file on the CD (one copy on the CD-ROM, and one in the setup executable).

2. Use the *Packages* dialog to create a package called "Hard Drive Install".

3. Select all the files on the CD-ROM tab, and use the *Multiple File Properties* dialog to assign them all to the "Hard Drive Install" package.

4. Use the *Screens* dialog to add a *Select Install Type* screen to your installer.

5. Configure the *Select Install Type* screen to offer two install types to the user, so they can choose to "Install to the hard drive" or "Leave files on the CD-ROM." The "Install to the hard drive" install type should have the "Hard Drive Install" package enabled, and the other install type should have it disabled.

6. Design the rest of your installer as you normally would.

   The end result should be that the user can choose whether to install all the files to their hard drive by selecting the appropriate install type on the *Select Install Type* screen.

Burning Your CD-ROM

Here are some guidelines that you should follow when preparing your CD-ROM:

- Burn your CD-ROM using the ISO-9660 format. This format will ensure that your CD is compatible with the vast majority of CD-ROM drives. Packet writing or "rewritable" formats are fine for testing purposes, but many of your users will not be able to access your CD if you use these formats.

- Consider making all files on your CD-ROM follow the "8.3" filename format. This is because the ISO-9660 format only supports these short filenames. If you use a long filename-supporting format like Joliet, your CD-ROM may not be compatible with older CD-ROM drives.

Creating an AutoPlay Menu

One nice final touch is to include an AutoPlay or "autorun" menu on your CD-ROM. An AutoPlay menu is a visual interface that is displayed automatically whenever the user inserts the CD into the CD-ROM drive.

The best way to create such a menu is to use a product designed for the purpose, such as Indigo Rose's AutoPlay Menu Studio. Please visit http://www.indigorose.com/autoplay for more details and a free evaluation version that you can download.

Chapter 18

Building and Distributing Your Installer

Building Your Installer

Once your installer has been configured and checked for accuracy, it is time to generate a setup executable.

To start the build process, select **Project | Build** from the menu. You can also start the build process by pressing the **Build** button (), or by using the F7 hotkey.

You will be asked to confirm that you want to start the build process. If you select **Yes**, the *Status* dialog will appear and the build process will proceed.

TIP

You can disable this confirmation step by deselecting the **Confirm before starting build process** check box on the General tab of the *Preferences* dialog. To access the *Preferences* dialog, select **Edit | Preferences** from the menu.

If setup executable already exists in the output folder, you will be asked to confirm that you want to overwrite the file.

NOTE

The output folder and setup executable filename can be configured on the Build Settings tab of the *Project Settings* dialog. To access the *Project Settings* dialog, select **Project | Settings** from the menu.

If all goes well, the setup executable will be generated in the output folder, ready for you to test and distribute.

TIP

You can start the build process from a batch file by running Setup Factory with the Unattended Build (/B) command line option. (See page 260.)

Testing Your Installer

Perhaps the most important and most often overlooked step when creating an installer is testing it after it has been built. You should test your setup executable on as many computers and operating systems as possible. Try it on Windows 95, Windows 98, and Windows ME. Try it on Windows NT, Windows 2000 and Windows XP. Try it using different screen resolutions, color depths and font sizes. Try it on systems with small amounts of hard drive space and on others with gigantic hard drives.

If you are supporting multiple languages with your installer, be sure to try running the setup executable in all of those languages. If you are distributing a lot of runtime files (such as the runtimes required by Visual Basic programs), try the install on a "virgin" system that does not have the runtime files installed, or on a base Windows 95A machine.

If you run into a problem with your installer, always try it on as many other systems as you can. You might be able to narrow down a common factor between the systems that is causing the installation (or your software) to fail.

Distributing Your Installer

Once your installer has been thoroughly tested, it is time to distribute your software. There are several ways to distribute your setup files, but the three main media are floppies, CD-ROMs, and the Internet.

CD-ROM Distribution

CD-ROMs have become the standard distribution media for software. Creating an installer appropriate for CD-ROM distribution is as simple as generating the setup executable and then burning the file onto a CD-ROM.

When building the setup executable, set the **Output folder** to be a directory on your hard drive, e.g. `C:\Output`, and choose "Largest possible" as the **Output file segment size**. This will create a single, self-contained `setup.exe` file. You can then simply burn this file to your CD-ROM just like any other file.

NOTE

If the setup executable is too large to fit on a single CD-ROM, set the output file segment size to the maximum file size that will fit on a CD. Setup Factory will create a series of files that you can then burn onto multiple discs. At run time, the user will be prompted to insert the next disc as required.

SEE ALSO

? For more information on CD-ROM installs, see page 247.

Internet Distribution

It is increasingly common to have setup executables available for download from the Internet. Creating a `setup.exe` file for Internet distribution is as simple as generating the setup executable and uploading it to your web site.

The instructions are the same as for building a CD-ROM distribution, but instead of burning the file to a CD, you will upload the file to your web site, and provide a download link to the file on your web page.

TIP

To avoid any problems with data corruption that might occur when users download the setup files, enable the **Verify archive before installing** option on the Settings tab of the *General Design* dialog .

Floppy Disk Distribution

Low-capacity floppy disks have almost been made obsolescent for software distribution by the rise of cheap CD-RW drives and recordable CDs. However, they still remain a popular media for very small distributions, if only for the near-universal availability of floppy drives. Setup Factory allows you to easily create multiple-floppy, "disk spanning" setup distributions.

The easiest way to prepare a floppy distribution is to build your setup executable right onto the floppies. Simply choose your floppy drive letter (e.g. "A:") as the **Output folder** on the Build Settings tab of the *Project Settings* dialog, and set the **Output file segment size** setting to "Largest possible." Setup Factory will automatically fill each floppy disk to its capacity and then ask for additional disks as required.

NOTE

Setup Factory will create a file called `setup.exe` on the first disk, `setup.2` on the second, `setup.3` on the third, etc.

Why not "1.44 MB Floppy?"
You may wonder why you wouldn't select "1.44 MB Floppy" as the **Output file segment size** when building the setup executable onto 1.44 MB floppies.

The "1.44 MB Floppy" setting is only provided in case you want to output the setup executable in 1.44 MB segments to a directory on your hard drive, and then copy the segments to floppy disks later.

Building your installer onto new, formatted floppies using the "Largest possible" setting is the easiest way to get the job done.

TIP

Always be sure to test your setup disks on several systems. This is especially true if you will be using the disks as masters to create duplicates.

Chapter 19

Command Line Options

Command line options are special values that can be passed to an executable file when it is run. Also known as "command line switches" or "arguments," command line options are usually used to set program options.

For example, entering "`C:\abc.exe /W /F`" on a command line would run a program called `abc.exe` and pass two command line options to it: `/W` and `/F`. The `abc.exe` program would see those options and handle them internally.

You can test command line options by running an executable from the Command Prompt in NT, or the DOS prompt in Windows. You can also use command line options in program shortcuts, or when running an application by using **Start -> Run**.

NOTE

Not all executables accept command line options, and the list of meaningful command line options is specific to each program. /W might mean "wait for return" in one program, but it could mean "enable wacky walk animation" in another—or it might not even be recognized at all.

TIP

Many executables will display a list of the command line options they support if you run them with the `/?` option.

Installer Options

The following command line options are supported by the Setup Factory installer:

Language (/L)

The `/L` option forces the installer to use the messages associated with a specific language ID, instead of using the messages that correspond with the user's system locale settings.

By default, the setup executable detects the language being used on the user's system, and uses the messages that were loaded from the corresponding language file at design time. If there were no messages defined for the user's language, the messages from the default language file are used.

SEE ALSO

? You can add different language files to your installer by using the Languages tab of the *General Design* dialog. For more information on the Languages tab, see page 130.

The /L option allows you to test your installer in multiple languages without having to change your system language and reboot.

The syntax for the /L option is:

```
/L:#
```

Replace # with the language ID for the language you want Setup Factory to "detect." The installer will configure itself as though that language was the current system language.

Example: `setup.exe /L:17`
(forces the installer to "detect" Japanese as the system language)

TIP

A complete list of language IDs can be found in the `C:\Program Files\Setup Factory\Languages\langids.ini` file.

Silent Mode (/S)

You can force the installer to run in silent mode by using the /S option. In silent mode, no screens, errors, or any other parts of the interface will be shown. This includes any messages displayed using the Show Message Box and Yes/No Message Box actions.

Example: `setup.exe /S`

NOTE

The built-in variable %SilentMode% is set to TRUE when the installer is running in silent mode.

Temp Path (/T)

Every setup executable requires some temporary space on the user's hard drive during the installation process. By default, Setup Factory uses the user's TEMP directory for extracting temporary files and other miscellaneous operations. You can force the setup executable to use an alternate directory by using the `/T` command line option.

The syntax for the `/T` option is:

`/T:`*path*

Replace *path* with the path to the folder you want the setup executable to use for its temporary files. (Be sure to put quotes around the entire argument if the path includes any spaces.) If the folder doesn't already exist on the user's system, it will be created automatically.

Example: `C:\Downloads\setup.exe "/T:C:\My Temp Dir"`
(forces the installer to use "C:\My Temp Dir" for temporary files)

Wait for Return (/W)

Use the /W option to have the Setup Factory launcher wait for the setup executable to return before exiting. This is useful if you're running the installer from another process and you want that process to wait for Setup Factory to finish before proceeding.

To keep the setup executable compact, part of its code is transported in a compressed form. Whenever the user runs your installer, the setup executable automatically extracts this compressed code, runs it, and exits—essentially handing the installation process over to the uncompressed code. By using the /W option, you can have the launcher program "stick around" until the end of the installation process.

Uninstaller Options

The following command line option is supported by the Setup Factory uninstaller:

Silent Mode (/S)

You can force the uninstaller to run in silent mode by using the /S option. In silent mode, no screens, errors, or any other parts of the interface will be shown. This includes any messages displayed using the Show Message Box and Yes/No Message Box actions.

Example: `iun600.exe "C:\Program Files\Foobar 2002\irunin.ini" /S`

NOTE

The built-in variable %SilentMode% is set to TRUE when the uninstaller is running in silent mode.

Design Environment (Build) Options

The following command line options are supported by the Setup Factory design environment:

Unattended Build (/B)

Performs an unattended build of a project. This allows you to build a setup executable "automatically" from a batch file.

The syntax for the /B option is:

/B:*unattended-build INI file*

Replace *unattended build INI file* with the path and filename of an INI file containing the unattended build settings you want Setup Factory to use. (Be sure to put quotes around the entire argument if the path or filename includes any spaces.)

Example: `SUF60Design.exe D:\foo.sf6 "/B:D:\release build.ini"`

NOTE

The project file name should always be passed as the first command line parameter.

The unattended-build INI file allows you to pass values into your project in the form of design-time constants. You can define as many design-time constants as you want in the INI file, with each constant on a separate line beneath the [Constants] section. For example:

```
[Constants]
#OUTPUTDIR#=C:\Output\Foobar 2002\Release
#SETUPNAME#=foobar2002setup.exe
#BUILD#=release
```

When you use the /B option, the specified project file is loaded into Setup Factory, the constants described in the specified unattended-build INI file are set, and the setup executable is generated—all without any interaction.

The SUF60Design.exe process returns an exit code of 1 if an error occurred during the unattended build, or 0 if the build was successful. You can use this return code to make your batch files respond to the success or failure of the Setup Factory build process.

TIP

The unattended build option is usually used in conjunction with the Minimize (/M) option.

Minimize (/M)

Minimizes the Setup Factory design environment when used with the unattended build (/B) option.

Example: `SUF60Design.exe C:D:\xyz.sf6 /B:D:\autobuild.ini /M`

Appendix A

Actions Index

The following actions are available in Setup Factory:

Action	*Category*	*Description*
Abort	*Control Structures*	Aborts the setup process.
Assign Value	*Variables*	Assigns a value to a variable.
Blank Line	*Control Structures*	Allows you to insert a blank line in an action list. Blank lines have no effect on run-time operation.
Call DLL Function	*Open/Close Programs*	Calls a specific function in a DLL file.
Check Internet Connection	*Internet*	Checks to see if the user's system is currently connected to the Internet.
Close Program	*Open/Close Programs*	Locates and closes a running program.
Comment	*Control Structures*	Allows you to insert a comment in an action list. Comments have no effect on run-time operation.
Continue Service	*Services*	Continues a previously paused Windows service.
Copy Files	*File Operations*	Copies files.
Count Delimited Strings	*Strings*	Counts the number of strings that could be made from an existing string by splitting it wherever specific delimiting characters are found.

Action	*Category*	*Description*
Count Text Lines	*Text Files*	Counts the number of lines in a text file.
Create Directory	*Folders*	Creates a directory.
Create Service	*Services*	Creates a Windows service.
Create Shortcut	*Shortcuts*	Creates a shortcut.
Delete File on Reboot	*Reboot*	Tells Windows to delete a file the next time the system is rebooted.
Delete Files	*File Operations*	Deletes files.
Delete Service	*Services*	Deletes a Windows service.
Delete Text Line	*Text Files*	Deletes the line of text at a specific (zero-based) line in a file.
Download File HTTP	*Internet*	Downloads a file from a web site.
END IF	*Control Structures*	Terminates an IF/END IF block.
END WHILE	*Control Structures*	Terminates a WHILE/END WHILE block.
Execute File	*Open/Close Programs*	Runs an executable file or batch file.
Find String	*Strings*	Searches for one string within another, and determines the location of the match if found.

Action	*Category*	*Description*
Format Number	*Strings*	Formats a floating-point number (i.e. real number) to a specified number of decimal places.
Generate Random Value	*Variables*	Generates a random integer value between two numbers, or generates a random string based on a mask (e.g. a serial number).
Get Delimited String	*Strings*	Gets a specific entry in a delimited string given delimiting characters and a zero-based index.
Get Text Line	*Text Files*	Gets the line of text at a specific (zero-based) line in a file.
GOTO Label	*Control Structures*	Jumps directly to a Label on the Actions tab.
IF	*Control Structures*	Begins an IF/END IF block. The actions in this block will only be performed if the condition in the IF action resolves to a non-zero (true) result.
Insert Text Line	*Text Files*	Inserts a line of text at a specific (zero-based) line in a file.
Install File	*File Operations*	Installs a file onto the user's system.
Label	*Control Structures*	Marks a position in the list of actions that can be reached using a GOTO Label action.
Left String	*Strings*	Creates a new string from the left-most *x* characters of an existing string.
Length of String	*Strings*	Counts the number of characters in a string.
Mid String	*Strings*	Creates a new string consisting of a number of characters starting from a given position in an existing string.
Modify INI File	*INI Files*	Sets an INI file value, deletes an INI file value, or deletes an INI file section.

Action	*Category*	*Description*
Modify Registry	*Registry*	Creates or deletes a Registry key or value.
Move File on Reboot	*Reboot*	Tells Windows to move a file the next time the system is rebooted.
Move Files	*File Operations*	Moves files between directories.
Open Document	*Open/Close Programs*	Opens, plays, prints or performs other actions on a document using its associated viewer.
Parse Path	*Strings*	Parses a string to extract specific information from it if it is a path.
Pause Service	*Services*	Pauses a Windows service.
Query Service	*Services*	Queries the system for the status of a specific service, and stores the status in a specified variable.
Read File Association	*File Information*	Assigns the path of the executable associated with a file extension (e.g. ".txt") to a variable.
Read File Information	*File Information*	Determines a file's version, its CRC value, its size in bytes, or whether it exists.
Read from INI File	*INI Files*	Reads a value from an INI file and stores it in a variable.
Read from Registry	*Registry*	Reads a value from the Registry and stores it in a variable, or sets a variable to True if a key exists.
Read Text File	*Text Files*	Reads the contents of a text file into a variable.
Register File	*File Operations*	Registers a DLL or OCX file with the user's operating system.

Action	*Category*	*Description*
Register Font	*File Operations*	Registers a TrueType font on the user's system.
Remove Directory	*Folders*	Removes (deletes) a directory on the user's system. The directory must be empty in order to be removed.
Remove Shortcut	*Shortcuts*	Removes (deletes) a shortcut file.
Rename File	*File Operations*	Renames a file.
Right String	*Strings*	Creates a new string from the right-most *x* characters of an existing string.
Run File on Reboot	*Reboot*	Tells Windows to execute a file the next time the system is rebooted.
Set File Attributes	*File Operations*	Sets the attributes of a file.
Show Message Box	*Dialogs*	Displays a standard Windows message box to present information to the user.
Start Service	*Services*	Starts a Windows service.
Stop Service	*Services*	Stops a Windows service.
Submit to Web	*Internet*	Submits data to a web site and stores the response. Allows you to perform a POST or GET to a web script or program just as you would from an HTML form.
Unzip Files	*File Operations*	Extracts all the files from a zip file to a directory on the user's system.
WHILE	*Control Structures*	Begins a WHILE/END WHILE block. The actions in this block will be performed while the condition in the WHILE action resolves to a non-zero (true) result.

Action	*Category*	*Description*
Write to Text File	*Strings*	Writes a string to a text file.
Yes/No Message Box	*Dialogs*	Displays a standard Windows message box to get a Yes/No response from the user. The response is stored in a variable that you specify.

Appendix B

Built-in Variables

%AppDir%	Your application's main directory, where all of your files and folders will be installed. For example, by default Setup Factory's main directory is: `C:\Program Files\Setup Factory 6.0`
%AppDrv%	The drive letter of %AppDir%.
%BDEPath%	The user's BDE (Borland Database Engine) directory.
%ColorDepth%	The color depth of the user's video display, in bits per pixel.
%CommonFiles%	The user's Common Files directory. Typically, this is something like: `C:\Program Files\Common Files`
%CompanyName%	Your company's name. The value of this variable is set in the **Company name** field on the Product Info tab of the *General Design* dialog.
%Copyright%	The copyright message for your product. The value of this variable is set in the **Copyright notice** field on the Product Info tab of the *General Design* dialog.
%CurrentDay%	A number representing the current day of the month, calculated when the setup begins.
%CurrentHour%	A number representing the current hour in 24-hour time (e.g. 4:00 PM is 16), calculated when the setup begins.

%CurrentMinute%	The current minute, calculated when the setup begins. This number is always expressed with two digits, so 4 minutes into the hour will be "04".
%CurrentMonth%	A number representing the current month, calculated when the setup begins. January is represented by "1" and December is represented by "12".
%CurrentSecond%	The current second, calculated when the setup begins. This number is always expressed with two digits, so 4 seconds into the minute will be "04".
%CurrentYear%	The four-digit number representing the current year, calculated when the setup begins.
%DAOPath%	The path to the user's DAO (Data Access Objects) directory.
%Date%	The current date on the user's system when the setup executable is run. It's in the format MM/DD/YY. For example, if the user ran the installer on May 23, 2002, %Date% would be: 05/23/02
%Desktop%	The path to the user's Desktop directory. On Windows NT, this is the path from the per-user profile.
%DesktopNT%	The path to the user's Desktop directory. On Windows NT, this is the path from the All Users profile.
%DoReboot%	Whether or not Setup Factory should restart the system at the end of the installation process. If set to any true value (e.g. "TRUE" or "1"), the system will be rebooted. Note: This variable is also used by the uninstaller. So, if %DoReboot% is set to "TRUE" during the uninstall, the user's system will be rebooted after the uninstall is complete.

%EuropeanDate%	The current date on the user's system when the setup executable is run, in the European dating format: DD/MM/YY. For example, if the user ran the installer on May 23, 2002, %EuropeanDate% would be: 23/05/02
%FontDir%	The path to the user's font directory.
%InfoURL%	The URL to a web site where the user can find more information about your product. The value of this variable is set in the **Information URL** field on the Product Info tab of the *General Design* dialog.
%ISODate%	The current date on the user's system when the setup executable is run, in the ISO format: YYYY-MM-DD. For example, if the user ran the installer on May 23, 2002, %ISODate% would be: 2002-05-23
%IsUserNTAdmin%	This variable is set to "True" if the user running the setup executable is currently logged into Windows NT, 2000 or XP with Administrator permissions. Otherwise, it's set to "False". On systems that aren't running some version of Windows NT, this variable is always set to "False".
%IsWin95%	This variable is set to "True" if the setup executable is running on Windows 95, and "False" otherwise.
%IsWin98%	This variable is set to "True" if the setup executable is running on Windows 98, and "False" otherwise.
%IsWinME%	This variable is set to "True" if the setup executable is running on Windows Millennium, and "False" otherwise.

%IsWinNT4%	This variable is set to "True" if the setup executable is running on Windows NT 4.0, and "False" otherwise.
%IsWin2000%	This variable is set to "True" if the setup executable is running on Windows 2000, and "False" otherwise.
%IsWinXP%	This variable is set to "True" if the setup executable is running on Windows XP, and "False" otherwise.
%JulianDate%	An integer value representing the number of days since midnight on January 1, 4713 B.C. Very useful when comparing dates or performing arithmetic to determine the number of days between two dates.
%LANDomain%	The domain that the user is logged in to. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
%LANHost%	The user's local computer name. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
%LANIP%	The user's IP address on the local network. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
%LANNIC%	The MAC address of the user's NIC *(*Network Interface Card). If the user's system does not contain a network card, this variable will default to "UNKNOWN".
%LANUser%	The user name that the user is currently logged in as. If the user's system is not connected to a LAN, this variable will default to "UNKNOWN".
%LastCommand%	The ID of the last action that was performed. (See the specific actions in the Command Reference for their respective action IDs.)

%LastErrorDetails%

If an error is generated, this variable is set to the "verbose" error message for the last action that occurred (the action that generated the error). When an action doesn't generate an error, this variable is empty ("").

For a complete list of the verbose error messages which can be generated by an action, see that action's documentation in the Command Reference.

%LastErrorMsg%

If an error is generated, this variable is set to the "simple" error message for the last action that occurred (the action that generated the error). When an action doesn't generate an error, this variable is empty ("").

For a complete list of the simple error messages which can be generated by an action, see that action's documentation in the Command Reference.

%LastErrorNum%

If an error is generated, this variable is set to an action-specific error number that identifies the last error that occurred. When an action does not generate an error, this variable is set to 0.

For a complete list of the error numbers associated with an action, see that action's documentation in the Command Reference.

%MousePresent%

Whether or not a user has a mouse connected to their system. This variable is set to "True" if the user has a mouse connected to their system; otherwise, it's set to "False".

%MyDocumentsDir%

The path to the user's My Documents directory, for example:

```
C:\My Documents
```

%NeedsReboot%	Whether or not the system needs to be restarted at the end of the installation process in order to install files that were in use. This is set to "True" when the system needs to be restarted, and "False" when the system doesn't.
%ProductName%	The name of the product that you are installing. The value of this variable is set in the **Product name** field on the Product Info tab of the *General Design* dialog.
%ProductTagline%	The marketing tagline for the product that you are installing. The value of this variable is set in the **Product tagline** field on the Product Info tab of the *General Design* dialog.
%ProductVer%	The version of the product that you are installing. The value of this variable is set in the **Version** field on the Product Info tab of the *General Design* dialog.
%ProgramFiles%	The path to the user's Program Files directory.
%RegOwner%	The name of the registered user of the system.
%RegOrganization%	The organization of the registered user of the system.
%SCFolderPath%	The full path to the shortcut folder on the Start menu where your application's shortcuts will be stored.
%SCFolderTitle%	The name of the shortcut folder on the Start menu where your application's shortcuts will be stored.
%ScreenHeight%	The user's screen height in pixels.
%ScreenWidth%	The user's screen width in pixels.
%SilentMode%	Whether or not the installer (or uninstaller) is running in silent mode. This variable is set to "True" when the installer (or uninstaller) is running in silent mode; otherwise, it's set to "False".

%SrcDir%	The full path to the folder the setup executable was run from. For example: `C:\Downloads`
%SrcDrv%	The drive that the setup executable was run from. For example: `C:`
%SrcFile%	The full path, including the filename, for the currently running setup executable. For example: `C:\Downloads\setup.exe`
%SoundCardPresent%	Whether there is a sound card installed on the user's system. Set to either "True" or "False".
%StartMenu%	The path to the user's Start menu directory. On Windows NT, this is the path from the per-user profile.
%StartMenuNT%	The path to the user's Start menu directory. On Windows NT, this is the path from the All Users profile.
%StartMenuPrograms%	The path to the Programs folder in the user's Start menu. On Windows NT, this is the path from the per-user profile.
%StartMenuProgramsNT%	The path to the Programs folder in the user's Start menu. On Windows NT, this is the path from the All Users profile.
%Startup%	The path to the user's Startup folder. On Windows NT, this is the path from the per-user profile.
%StartupNT%	The path to the user's Startup folder. On Windows NT, this is the path from the All Users profile.

%SysDir%	The path to the user's Windows System directory, for example: `C:\Windows\System`
%SysDrv%	The drive that the user's Windows System directory is located on. For example: `C:`
%SysLanguage%	The numeric primary language ID for the user's system language.
%SystemRAM%	The amount of physical memory on the user's system in megabytes.
%TempDir%	The path to the user's Temp directory.
%TempLaunchDir%	The path to the temporary directory where Setup Factory extracts the files it will need for the installation. (For example, this is the directory where Primer files are extracted to.) Usually this directory will be the user's temporary directory, unless the user overrides the temporary directory with the /T command line option.
%WinDir%	The path to the user's Windows directory. For example: `C:\Windows`

Design-time Constants

Design-time constants are similar to variables, but instead of being converted to values at *run* time, design-time constants get converted at *build* time. We call them *design*-time constants because the names you give them only exist at design time. At build time, the name of each design-time constant is replaced by the value that was assigned to it.

Here are some design-time constants that are built into Setup Factory:

#ASC???#	This is a special set of design-time constants. When a constant's name starts with "#ASC", Setup Factory will interpret the value assigned to that constant as the ASCII value of a character, and will replace the constant's name with that character at build time. So, for example, to create a design-time constant to represent a null character (ASCII value 0), you could assign the value "0" to "#ASCNULL#".
#JULIANDATE#	An integer value representing the number of days since midnight on January 1, 4713 B.C. Very useful when comparing dates or performing arithmetic to determine the number of days between two dates. Note: #JULIANDATE# is a design-time constant. This means that it will equal the date *when your setup executable was built*.
#PROJECTFILEDIR#	The path to the directory where your Setup Factory project (`.sf6`) file was saved.
#SUFDIR#	The path to the directory where Setup Factory is installed on your development system.
#SYSDIR#	The path to the SYSTEM directory on your development system.
#WINDIR#	The path to the Windows directory on your development system.

Appendix C

Contact Info

Indigo Rose Corporation is a world leader in software installation and deployment tools. Programmers, electronic publishers, multimedia developers and software professionals from all over the world turn to Indigo Rose Corporation for state-of-the-art solutions.

Corporate Headquarters

Web:	http://www.indigorose.com
Email:	info@indigorose.com
Office Hours:	Monday to Friday 9:00 AM to 5:00 PM Central Standard Time

Sales

The Indigo Rose sales team is ready to answer your questions Monday to Friday from 9:00 AM to 5:00 PM Central Standard Time.

Contact a sales representative for information on the latest Indigo Rose products or to purchase technical support subscriptions, upgrades or additional product licenses.

If your question is technical in nature, please contact Technical Support.

Phone:	(204) 946-0263
Fax:	(204) 942-3421
Web:	http://www.indigorose.com
Email:	sales@indigorose.com

Technical Support

Indigo Rose Corporation offers a variety of technical support programs designed to match your specific needs. Both complimentary and fee-based support options are available. Please refer to the *Developer Support Programs* brochure included with your package for complete details on the available support programs.

Your connection to all of our support resources can be found at:
http://www.indigorose.com/support/

Before You Contact Our Support Department

Save yourself both time and money by referring to our supplementary resources *before* you contact our technical support department. Answers to just about any question can be found in these self-help resources. You'll be able to get the answers you need 24 hours a day, 7 days a week.

Product Documentation

The product documentation includes this User's Guide, the Command Reference, and any last-minute notes in a readme.txt file. Answers to most common support issues can be found in the product documentation.

Knowledge Base

A collection of informational articles and how-to tutorials. These articles cover both common and uncommon situations, including advanced technical issues. Many articles are written in response to consulting-type situations. You'll find a wealth of good information in the knowledge base.

Discussion Groups

Indigo Rose maintains a number of community message boards. These discussion groups are frequented by many developers, including our own technical support representatives. The discussion groups are a great source for ideas, solutions and peer support.

Limitations of Technical Support

Technical Support is limited to general product usage questions. Questions relating to external functionality such as operating systems, development environments, third-party products or various Windows technologies are not included. All services are provided subject to the current *Support Terms and Conditions* as listed at our web site.

Limitations include, but are not limited to:

- Problems with your development tool such as Visual Basic, Delphi, Director, etc. Please consult your development tool's documentation for these issues.
- Third-party technologies you are distributing (such as BDE or QuickTime).
- Issues relating to distribution media such as CD-R burning.
- A specific design issue that is unique to your distribution.

TIP

If your situation requires a more personalized solution, you might want to consider using our Consulting Services. Our consulting department can help you with matters that go beyond standard technical product issues.

For more information on Consulting Services, please visit our web site.

Appendix D

Minimum System Requirements

Setup Factory requires the following minimum system configuration in order to operate properly:

Setup Factory Design Environment

- Windows 95 (OSR2), 98, ME, NT 4.0 (SP3), 2000, XP
- Pentium processor
- 128 MB RAM
- 800x600 SVGA display with small fonts setting enabled
- Video card set to 16 bit 32K color or greater (recommended)
- Mouse
- 20+ MB free hard drive space

Setup Factory Run-Time Executable

- Windows 95, 98, ME, NT 4.0 (SP3), 2000, XP
- 486 processor
- 32 MB RAM
- 640x480 SVGA display
- Video card set to 8 bit 256 color or greater (recommended)
- 10+ MB free hard drive space

Glossary

action An instruction that the Setup Factory installer can perform at run time.

application An executable program capable of performing several tasks or functions.

associativity A way of describing the rules that determine which sub-expressions are performed first when operators have equal precedence. If two operators have the same precedence and are left-associative, the operation on the left is performed first. If two operators have the same precedence and are right-associative, the operation on the right is performed first. (Most of the operators in Setup Factory are left-associative.) [*See: expression, operator, precedence, sub-expression*]

binary operator An operator that operates on two operands. In Setup Factory, all binary operators are infix operators, so one operand precedes the operator, and the other operand follows it. In other words, a binary operator operates on one value to the left of it, and one value to the right. [*See: expression, operand, operator*]

Boolean A term for values that are used to describe the results of logical comparisons. A Boolean value can be either True or False.

build time When Setup Factory generates the setup executable.

built-in variable A variable automatically defined for you by Setup Factory. [*See: variable*]

CRC value *Cyclic Redundancy Check* value. A 32-bit checksum number calculated from the contents of a file, that changes whenever the file's contents change.

current release The most up-to-date external (meaning "available to users") version of your software. Also called the "latest release."

custom variable A variable that you define yourself. [*See: variable*]

design time The process of designing your installer using the Setup Factory design environment.

design-time constant A special kind of variable that gets converted at build time instead of at run time. They are called design-time constants because the names you give them only exist at design time. When Setup Factory builds the setup executable, the name of each design-time constant is replaced by the value you assigned to it.

In Setup Factory, design-time constant names always begin and end with a number sign (#). [*See: variable*]

dialog A window that displays options or questions in order to receive input or instruction from a user. Also: a single window in an interface made up of logical steps that must be followed. [*See: wizard*]

directory A named location within a file system where other folders or files can be stored. [*See: folder, file*]

drive A form of fixed, networked or removable media used as a storage device. Also: the letter assigned to a drive during the computer boot process.

executable A program file, usually containing instructions for a computer in the form of machine code. In Windows, executables typically have a `.exe` extension.

expression Any valid combination of operands and operators that resolves to a single result. For example, the expression (4 + 3) * 2 resolves to 14. [*See: operand, operator*]

extension A period (.) followed by one or more letters at the end of a filename.

Windows uses the file extension to determine what kind of information is contained in a file. For example, in the filename `myfile.txt`, `.txt` is the file extension that identifies `myfile.txt` as a text file.

external version A version of your software that is available to users. [*See: release, version*]

file A collection of data stored as a single entity in a file system.

Some files only serve as receptacles for data (text files, document files, bitmap image files), while others contain instructions for the system to perform (program files, dynamically linked library or "`.dll`" files).

filename The name given to a file when it is created or saved.

firewall A firewall is a combination of computer hardware and software used to keep a network secure. The firewall acts as a gatekeeper between an internal network and the Internet, allowing only specific kinds of messages to flow in or out.

folder The common term for a directory in Windows. A named location within a file system where other folders or files can be stored. [*See: directory*]

full-history patch An advanced patch that can update all older versions of a software product using a single self-extracting executable.

Indigo Rose's Visual Patch creates full-history patches. [*See: patch*]

HTTP *Hypertext Transfer Protocol.* The Internet protocol used to transmit and receive data over the World Wide Web. Often used between a Web browser and a server to request a document and transfer its contents.

infix operator An operator that needs to be placed between the operands it operates on. For example, to multiply 2 by 3 with the Multiply operator (which is a binary infix operator) you would place the Multiply operator between the two values, like so: 2 * 3. [*See: binary operator, operand, operator*]

INI file Short for "initialization file." A text file where user, application or system settings can be stored and retrieved by an application or the system as required.

internal version A version of your software that is not available to users. In other words, a version of your software that is never released, but only exists internally. [*See: version*]

Intranet An "internal Internet" designed for use within a single company, university or organization. Essentially, a private network using the same technologies that drive the Internet. The main difference between an intranet and the Internet is that an intranet is not meant to be accessible to the public.

Julian Date In Setup Factory, this is a date expressed as an integer using the Chronological Julian Day numbering system. This system counts the number of days since midnight on January 1, 4713 B.C.

The Julian Date is primarily useful when comparing dates or performing arithmetic to determine the number of days between two dates.

LAN *Local Area Network*. A network that connects computers located on the same floor or in the same building or nearby buildings.

latest release The most up-to-date external (meaning "available to users") version of your software. Also called the "current release."

NIC *Network Interface Card*. A computer component that allows the computer to physically connect to and communicate over a network.

operand A value or sub-expression that an operator operates on. For example, in the expression 3 + 4, the values "3" and "4" are both operands of the "+" (or "Add") operator.
[*See: expression, operator*]

operator A symbol used to represent an operation that can be performed on one or more operands in an expression. For example, in the expression 3 + 4, the "+" is the symbol for the "Add" operator.
[*See: expression, operand*]

password A string of text used to gain access to something.
[*See: serial number*]

patch A file that, when run, modifies or replaces specific files on a computer system, usually to bring an already-installed software product up to date.
[*See: full-history patch*]

path Text that describes a location within a file structure. Each path describes a route followed by the operating system to find, store or retrieve a file or folder.

In Windows and MS-DOS, each folder in a path is separated by a backwards slash (\).

portable executable file An executable file that is portable across all 32-bit Microsoft operating systems. Each Portable Executable (or "PE") file will run on all versions of Windows 95, 98, ME, NT, 2000 and XP.

precedence A way of describing the rules that determine which operators in a series of sub-expressions are applied first. Operators with higher precedence are applied before operators with lower precedence.
[*See: associativity, expression, operator, sub-expression*]

prefix operator An operator that needs to precede the operand it operates on. For example, to make the value 2 a negative number with the unary minus operator (which is a prefix operator), you would place the unary minus operator before the value, like so: -2.
[*See: infix operator, operand, operator, unary operator*]

protocol A set of rules that computers use to communicate with each other. Protocols ensure that different kinds of network hardware and software can work together.

Registry A database used by 32-bit versions of the Windows operating system to store system and software configuration details.

The Windows Registry serves a similar purpose to the `win.ini` and `system.ini` files used by earlier versions of Windows.

release A set of files that are distributed as a whole unit, i.e. all the files that make up one version of your software.
[*See: external version, version*]

root folder The "base" or "main" folder on any drive, where all of the other folders on a drive are located.

When you double-click on the C: drive in My Computer, you're opening the root folder of the C: drive

run time When the actual setup executable is run.

screen A window in a graphical user interface.

The user interface of a Setup Factory installer is made up of individual screens which can be added to or removed from any Setup Factory project. [*See: dialog, window*]

serial number Some text (usually a sequence of numbers and/or letters) used to enable or identify one instance of a software product.

server A computer on a network that runs programs and stores data for use by other computers. Servers store information and respond to requests for that information by "serving" the information to other computers.

service An application type supported by Windows NT, 2000 and XP that conforms to the interface rules of the Service Control Manager (SCM). Services can be started programmatically by applications using Windows API function calls, manually by users via the Services applet in the Control Panel, or automatically by the Service Control Manager during the system boot process.

Because services can run when there is no user logged into the system, it is often desirable to run an application as a service (if it conforms to the SCM interface rules) so it can perform operations before any user logs on, and so it can continue to operate after the current user logs out of the system.

shell An interface between a user and an operating system. Often used to present an alternative interface which abstracts the details of the operating system and allows a user to perform tasks without accessing the underlying operating system directly. [*See: shell operation*]

shell operation An operating system task performed by the shell, usually at the request of a user or application. [*See: shell*]

shortcut [*See: shortcut file*]

shortcut file A very small file that links to another file or web page in the Windows operating system. Items in the Favorites menu and the Start menu are all shortcuts. (Shortcut files have a `.lnk`, `.pif`, or `.url` extension that is hidden by the Windows operating system.)

shortcut folder A folder that contains shortcut files.

shortcut icon A visual representation of a Shortcut File in Windows. [*See: shortcut file*]

string A series of ASCII characters. This can be a word, phrase, number, or an entire book.

sub-directory A directory that is located within another directory. [*See: sub-folder, folder, directory*]

sub-expression An expression used as an operand in a larger expression. For example, (1 + 2) * (3 - 1) is an expression made up of two sub-expressions, "(1 + 2)" and "(3 - 1)". [*See: expression, operand*]

sub-folder A folder that is located within another folder. [*See: sub-directory*]

unary operator An operator that operates on a single operand. The unary operators in Setup Factory are all prefix operators, which means they precede their operands. In other words, a unary operator operates on the value to the right of it. [*See: binary operator, expression, operand, operator*]

UNC path A Universal Naming Convention path. The Universal Naming Convention is a way to refer to files, folders and volumes in a device-independent way (especially across a network).

The format for a UNC path is:
`\\server\volume\directory\file`

variable A special named "container" for values that change.

In Setup Factory, variable names always begin and end with a percentage sign (%).

version A single instance or variant of something that has changed or is expected to change over time. When changes are made to a software application, the result is a new "version" of the original application.

Also, the name or number used to identify one version from another. [*See: external version, internal version, release*]

WAN *Wide Area Network*. A network that connects computers over long distances. A WAN connects computers that are physically or even geographically far apart.

window A rectangular area used as a canvas in the Windows operating system's graphical user interface, upon which objects such as text or buttons can be "drawn" by the operating system or individual applications. These windows can often be manipulated separately (moved, resized, minimized, maximized, etc.).

wizard An interface made up of logical steps that must be followed, presented using successive dialog windows, often including additional explanation or involving simplification to streamline steps or be friendlier to novice users. [*See: dialog, screen*]

Index