

# Software Packager User's Guide

Document Number 007-2503-003

## CONTRIBUTORS

Written by Martha Levine and Terri Wanke

Production by Ruth Christian

Engineering contributions by Kirk Erickson and David Fenstermaker, and Richard Wright

Cover design and illustration by Rob Aguilar, Rikk Carey, Dean Hodgkinson, Erik Lindholm, and Kay Maitz

© Copyright 1996, Silicon Graphics, Inc.— All Rights Reserved

The contents of this document may not be copied or duplicated in any form, in whole or in part, without the prior written permission of Silicon Graphics, Inc.

## RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure of the technical data contained in this document by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and/or in similar or successor clauses in the FAR, or in the DOD or NASA FAR Supplement. Unpublished rights reserved under the Copyright Laws of the United States. Contractor/manufacturer is Silicon Graphics, Inc., 2011 N. Shoreline Blvd., Mountain View, CA 94043-1389.

Silicon Graphics and IRIS are registered trademarks, and IRIS and Indigo Magic are trademarks of Silicon Graphics, Inc.

---

# Contents

**List of Figures** vii

**List of Tables** ix

**About This Guide** xi

What This Guide Contains xi

What You Should Know Before Reading This Guide xii

Font Conventions in this Guide xii

**1. Packaging Software for Installation: An Overview** 3

About *swpkg* 3

Running *swpkg* 4

Using the *swpkg* Worksheets 4

Using the *swpkg* Menus 5

    Using the File Menu 5

    Using the View Menu 7

    Using the Help Menu 7

Using the Worksheet Selection Tabs 8

Using the Message Area 8

**2. The *swpkg* Tutorial** 13

Step One: Setting Up the Example Application Used by This Tutorial 13

Step Two: Running *swpkg* 14

Step Three: Creating a Product Hierarchy 16

Step Four: Tagging the Files 18

Step Five: Setting Permissions and Destinations 23

Step Six: Adding Attributes 27

Step Seven: Building the Product 31

Step Eight: Installing and Running the Product 33

- 3. **Creating a Product Hierarchy** 37
  - Creating a Product Hierarchy: Before You Begin 37
    - Prerequisites 37
    - About Product Hierarchies 38
    - What's a Spec File? 40
  - Creating a Product Hierarchy: The Basic Steps 41
  - Using the Create Product Hierarchy Worksheet 42
    - Creating a Product Structure 44
    - Entering Product Specifications 47
    - Entering Image Specifications 50
    - Entering Subsystem Specifications 55
    - Editing Mapping Expressions Directly 64
    - The *Assign* Arrow Button 64
  
- 4. **Tagging the Files** 69
  - Tagging the Files: Before You Begin 69
    - Prerequisites 69
    - What's an IDB File? 69
    - What's a Tag? 70
  - Tagging the Files: The Basic Steps 71
  - Using the Tag Files Worksheet 72
    - Selecting Product Files Using the File Browser 73
    - Accessing Your IDB File Using the IDB File Viewer 75
    - Selecting Tags Using the Tags Browser 78
  
- 5. **Editing Permissions and Destinations** 83
  - Editing Permissions and Destinations: Before You Begin 83
    - Prerequisites 83
    - What Are Permissions and Destinations? 84
  - Editing Permissions and Destinations: The Basic Steps 84
  - Using the Edit Permissions & Destinations Worksheet 85
    - Setting Source Tree Roots 86
    - Setting Permissions and Destination Directories 88
    - Resetting All Text Fields to the Default Values 91

- 6. **Adding Attributes** 95
  - Adding Attributes: Before You Begin 95
    - Prerequisites 95
    - What Are Attributes? 96
  - Adding Installation Attributes: The Basic Steps 96
  - Using the Add Attributes Worksheet 97
    - Selecting Software Installation Attributes 98
    - Selecting Hardware Installation Attributes 105
    - Assigning the Selected Attributes 108
- 7. **Building the Product** 111
  - Building the Product: Before You Begin 111
    - Prerequisites 111
    - How Does *swpkg* Build a Product? 112
  - Building the Product: The Basic Steps 112
  - Using the Build Product Worksheet 113
    - The Spec File Path Label 115
    - Setting Tree Root and the Distribution Directory 115
    - Selecting Build Options 116
    - Running a Test Build 118
    - Building the Product 118
  - After the First Build 118
    - Building the Product After the First *swpkg* Build 119
    - Combining Existing Products Into a Single Product 121
    - Incorporating the Help Subsystem into a Product 123
    - Breaking an Existing Product Into Two Products 123
- 8. **Creating a Patch Product** 127
  - Creating a Patch Product: The Basic Steps 127
    - Patch Product Requirements and Concepts 128
- A. **Writing Mapping Expressions** 133
  - About Mapping Expressions 133
  - Variables and Data Types 134
  - Operators 134

	Built in Variables	135
	Built-In Functions	136
	Statements	137
	Example	137
<b>B.</b>	<b>Troubleshooting</b>	141
	Checklist of Do's and Don'ts	141
	Error Messages	142
	Other Problems	142
	<b>Index</b>	145

---

## List of Figures

<b>Figure 1-1</b>	The <i>swpkg</i> File Menu	5
<b>Figure 1-2</b>	The <i>swpkg</i> View Menu	7
<b>Figure 1-3</b>	The <i>swpkg</i> Help Menu	7
<b>Figure 1-4</b>	The Worksheet Selection Tabs	8
<b>Figure 1-5</b>	The Message Area	9
<b>Figure 2-1</b>	The Create Product Hierarchy Worksheet	15
<b>Figure 2-2</b>	The Completed Product Hierarchy Worksheet	17
<b>Figure 2-3</b>	The Tag Files Worksheet	19
<b>Figure 2-4</b>	The Completed Tag Files Worksheet	22
<b>Figure 2-5</b>	The Edit Permissions & Destinations Worksheet	24
<b>Figure 2-6</b>	The Completed Worksheet for the <i>Finance.ad</i> File	26
<b>Figure 2-7</b>	The Add Attributes Worksheet	28
<b>Figure 2-8</b>	The Completed Add Attributes Worksheet	30
<b>Figure 2-9</b>	The Build Product Worksheet	31
<b>Figure 3-1</b>	Example Product Hierarchy	39
<b>Figure 3-2</b>	Example Product Hierarchy Names	40
<b>Figure 3-3</b>	The Create Product Hierarchy Worksheet	43
<b>Figure 3-4</b>	The Product Hierarchy Graph	44
<b>Figure 3-5</b>	Graph Display Controls	46
<b>Figure 3-6</b>	The Product Specification Sheet	48
<b>Figure 3-7</b>	The Image Specification Sheet	51
<b>Figure 3-8</b>	The Subsystem Specification Sheet	55
<b>Figure 4-1</b>	The Tag Files Worksheet	72
<b>Figure 4-2</b>	The File Browser	73
<b>Figure 4-3</b>	The IDB File Viewer	76
<b>Figure 4-4</b>	The Tags Browser	79
<b>Figure 5-1</b>	The Edit Permissions & Destinations Worksheet	86

---

<b>Figure 5-2</b>	The Root Specification Sheet	87
<b>Figure 5-3</b>	The Permissions and Destinations Sheet	88
<b>Figure 6-1</b>	The Add Attributes Worksheet	97
<b>Figure 6-2</b>	The Attributes Specification Sheet: Software Attributes	98
<b>Figure 6-3</b>	The Attributes Specification Sheet: Hardware Attributes	106
<b>Figure 7-1</b>	The Build Product Worksheet	114



---

## List of Tables

<b>Table 2-1</b>	Permissions and Destinations for Remaining Files	27
<b>Table 6-1</b>	Configuration Types	99



---

## About This Guide

This book describes how to use Software Packager (*swpkg*), a graphical tool for packaging software for installation on Silicon Graphics<sup>®</sup> workstations. Products packaged with Software Packager can be installed with Software Manager (*swmgr*), an Indigo Magic<sup>™</sup> Desktop utility for installing software.

### What This Guide Contains

This book contains the following chapters:

Chapter 1, "Packaging Software for Installation: An Overview," provides an overview to Software Packager.

Chapter 2, "The swpkg Tutorial," contains a brief tutorial of packaging an application using Software Packager.

Chapter 3, "Creating a Product Hierarchy," describes how to create a product hierarchy.

Chapter 4, "Tagging the Files," describes how to tag files for inclusion in the product and assign the files to subsystems in your product.

Chapter 5, "Editing Permissions and Destinations," describes how to set the destination, names, and permissions of the files installed.

Chapter 6, "Adding Attributes," describes how to set attributes for the files installed, including how to execute commands before and after installation, how to modify configuration files, and how to specify which files to install on different hardware configurations.

Chapter 7, “Building the Product,” shows how to build the installable product. It also discusses how to change a product that you’ve built, how to merge two or more existing products into a single product, and how to divide an existing product into two or more separate products.

Chapter 8, “Creating a Patch Product,” shows how to package a patch product.

Appendix A, “Writing Mapping Expressions,” describes how to create optional mapping expression for complex cases of assigning files to subsystems.

Appendix B, “Troubleshooting,” provides tips for troubleshooting common Software Packager problems.

## What You Should Know Before Reading This Guide

This guide assumes that you are familiar with installing software using Software Manager. For more information on Software Manager and software installation, see the *Personal System Administration Guide* and the *Software Installation Administrator’s Guide*.

Silicon Graphics provides both these manuals online. You can view them from the IRIS InSight Viewer. To use the IRIS InSight Viewer, select “On-line Books” from the Help toolchest.

## Font Conventions in this Guide

These style conventions are used in this guide:

- **Boldfaced text** indicates that a term is an option flag, a data type, a keyword, a function, or an X resource.
- *Italics* indicates that a term is a file name, a button name, a variable, an IRIX command, a document title, or an image or subsystem name.
- “Quoted text” indicates menu items.

- `Screen type` is used for code examples and screen displays.
- **screen type** is used for user input and nonprinting keyboard keys.
- Regular text is used for menu and window names.



## **Packaging Software for Installation: An Overview**

This chapter provides an overview to using Software Packager to create installable products. It introduces basic concepts and terminology.





# Packaging Software for Installation: An Overview

This chapter provides a brief overview of Software Packager (*swpkg*), a graphical tool for packaging software for installation on Silicon Graphics workstations. It contains these sections:

- “About *swpkg*” on page 3 provides an overview of *swpkg* and the Silicon Graphics installation process.
- “Running *swpkg*” on page 4 describes how to run *swpkg*.
- “Using the *swpkg* Worksheets” on page 4 gives an overview of the *swpkg* worksheets.
- “Using the *swpkg* Menus” on page 5 describes the *swpkg* menus.
- “Using the Worksheet Selection Tabs” on page 8 describes how to select *swpkg* worksheets.
- “Using the Message Area” on page 8 describes the use of the *swpkg* message area.

**Note:** To package your application for installation, you must install the *inst\_dev* product, which contains *swpkg*, related utilities, and reference pages.

## About *swpkg*

The *swpkg* application allows you to quickly and easily package your application for Software Manager (*swmgr*(1M)), the Silicon Graphics software installation program. *swpkg* takes your product’s files and, using the additional information that you enter into the *swpkg* worksheets, creates compressed software images, readable by Software Manager. *swpkg* also creates the two other files required by Software Manager, called the *spec* and *IDB* files. (*spec* and *IDB* files are the files in which *swpkg* stores all the information about your product installation that you enter into the *swpkg*

worksheets. For a definition of spec files, see “What’s a Spec File?” on page 40. For a definition of IDB files, see “What’s an IDB File?” on page 69).

## Running *swpkg*

Invoke *swpkg* by typing:

```
swpkg
```

## Using the *swpkg* Worksheets

To use *swpkg*, you enter information about your product in a series of worksheets. The *swpkg* window always displays one of the five available worksheets. To see a worksheet other than the one currently displayed, use the Worksheet Selection tabs, discussed in “Using the Worksheet Selection Tabs” on page 8.

Here are the five *swpkg* worksheets, listed in the order in which you should fill them out:

### Create Product Hierarchy

“Creating a Product Hierarchy: The Basic Steps” on page 41 lists the steps for filling out this worksheet.

### Tag Files

“Tagging the Files: The Basic Steps” on page 71 lists the steps for filling out this worksheet.

### Edit Permissions and Destinations

“Editing Permissions and Destinations: The Basic Steps” on page 84 lists the steps for filling out this worksheet.

### Add Attributes

“Adding Installation Attributes: The Basic Steps” on page 96 lists the steps for filling out this worksheet.

### Build Product

“Building the Product: The Basic Steps” on page 112 lists the steps for filling out this worksheet.

Work through the worksheets in order (beginning with the Create Product Hierarchy worksheet, which is the worksheet that’s automatically selected when you start up *swpkg*). Fill in all necessary information before moving to the next worksheet. (It’s necessary to work through the worksheets in order

because the later worksheets depend on information entered in the earlier worksheets.)

If you're new to packaging for Software Manager, you might want to read Chapter 2, "The swpkg Tutorial," before you try to use *swpkg* to package your own application.

## Using the swpkg Menus

*swpkg* has three pull-down menus located on the menu bar, the horizontal bar at the top of the *swpkg* window.

The three pull-down menus are:

- The File Menu, which allows you to clear, open, save, and append spec and IDB files or to quit *swpkg*
- The View Menu, which allows you to hide or show the Message Area
- The Help Menu, which provides your main entry point into *swpkg*'s online help

The rest of this section explains each of the three menus in more detail.

### Using the File Menu

The File pull-down menu, shown in Figure 1-1, provides menu items that allow you to clear, open, save, and append spec and IDB files or to quit *swpkg*. (spec and IDB files are the files in which *swpkg* stores all the information about your product installation that you enter into the *swpkg* worksheets.)

Each item on the File pull-down menu (except "Quit") has a rollover menu that allows you to choose whether you want the selected action performed on the spec file, the IDB file, or both. All of the File menu items log a message in the Message Area (located below the Worksheet Selection tabs).

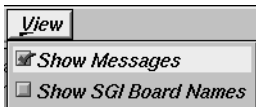
**Note:** When creating or renaming spec and IDB files, make sure that your spec filenames end with the *.spec* suffix and IDB filenames end with the *.idb* suffix.



**Figure 1-1** The *swpkg* File Menu

Here's a brief description of each of the items on the File pull-down menu:

- Use the "New" menu item to clear the spec and or/IDB file(s) for the current *swpkg* session of all entries. This overwrites any existing spec and/or IDB files for the current session. (*swpkg* posts a dialog asking if you want to abandon the current spec and IDB files before opening new ones. If you click the *Cancel* button, *swpkg* aborts creating a new spec and/or IDB file(s).)
- Use the "Open" menu item to open an existing spec and/or IDB file and load the information from the opened file(s) into the current *swpkg* session. This overwrites any existing spec and/or IDB files for the current session. (*swpkg* posts a dialog asking if you want to abandon the current spec and IDB files before opening new ones. If you click the *Cancel* button, *swpkg* aborts creating a new spec and/or IDB file(s).)
- Use the "Append" menu item to append an existing spec and/or IDB file onto the corresponding file(s) for the current *swpkg* session.
- Use the "Save" menu item to save the current spec file, IDB file, or both. These files contain the information you've entered into *swpkg*'s worksheets during the current session. If you haven't yet opened or saved the spec and/or IDB file(s) for the current product, *swpkg* asks you to specify the filename(s).
- Use the "Save As" menu item to save all the information you've entered into the current *swpkg* session. (This information is saved in the spec and IDB files.) When you select the "Save As" menu item, brings up a file selection box for specifying an arbitrary path and filename for your files.
- Use the "Create Patch..." menu item to begin the process of creating a patch product. When you select this item, a dialog opens in which you name your patch product and select the files you want to include in the product. Before using this item, you need to "Open" the spec and IDB files for the original product. See Chapter 8, "Creating a Patch Product."
- Use the "Quit" menu item to quit *swpkg*.



**Figure 1-2** The *swpkg* View Menu

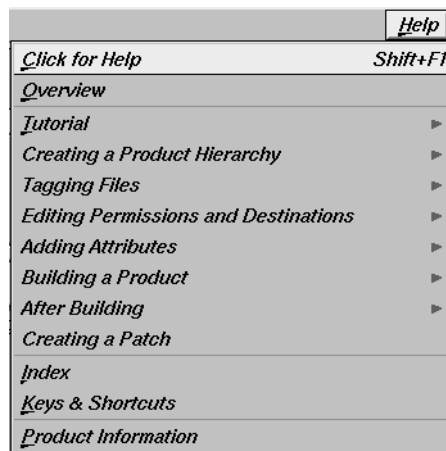
## Using the View Menu

The View pull-down menu, shown in Figure 1-2, contains “Show Messages” and “Show SGI Board Names.” Here’s a brief description:

- Check the “Show Messages” menu item when you want the Message Area (shown in Figure 1-5) is displayed. When an important message is logged, the Message Area appears automatically.
- Check the “Show SGI Board Names” when you want the board names displayed in the Add Attributes worksheet.

## Using the Help Menu

The Help pull-down menu, shown in Figure 1-3, provides easy access to *swpkg*’s online help.



**Figure 1-3** The *swpkg* Help Menu

To display help:

- Select the “Click for Help” menu item to get help on a specific area of a worksheet. When you select “Click for Help” the cursor turns into a question mark. Position this question-mark cursor over the area that you want information about, then click the left mouse button. A help

card appears. If you prefer, you can get the question-mark cursor by holding down the <Shift> key and pressing the <F1> key.

- Select the “Overview” menu item to see a brief overview of *swpkg*.
- Select the “Tutorial” menu item to see the *swpkg* tutorial.
- Select any of the listed help topics for help on that topic.
- Select the “Index” menu item, to see a list of available help cards.
- Select the “Keys and Shortcuts” menu item, to see a list of *swpkg*’s keys and shortcuts.
- Select the “Product Info” menu item to see a brief product information message.

## Using the Worksheet Selection Tabs

Use the five Worksheet Selection tabs, shown in Figure 1-4, to work through *swpkg*’s five main worksheets. You can view only one worksheet at a time. Access each worksheet by clicking the left mouse button on the tab associated with that worksheet.

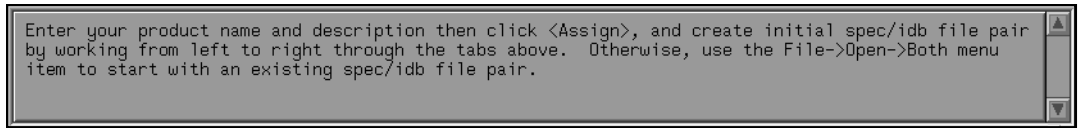


**Figure 1-4** The Worksheet Selection Tabs

You can tell which worksheet you’re in by looking at which tab is selected. In Figure 1-4, the Create Product Hierarchy tab is selected.

## Using the Message Area

The Message Area, shown in Figure 1-5, is a scrollable text area containing messages from *swpkg*. In particular, the Message Area shows you error messages and provides feedback during the build process.



**Figure 1-5** The Message Area

If you prefer not to see this window, use the View pull-down menu on the menu bar to hide the Message Area.





## **The *swpkg* Tutorial**

After going through this brief tutorial, you should understand the basic features of *swpkg* and how to use it to create an installable product.



## The *swpkg* Tutorial

This chapter provides a brief (ten-minute) tutorial that takes you through the basic steps for installing a simple product. When you've finished, you'll have packaged an example application for installation. It's best to work along with the tutorial, completing the steps on your own workstation as you read through them.

**Note:** *swpkg* is a versatile tool that allows you a great deal of freedom in creating installable files. Many of *swpkg*'s features are not required to package an average product for installation. Rather than present all the features, this tutorial focuses on the basics needed to package an application. Don't worry that we don't use some of the buttons and text entry fields that appear in the worksheets. *swpkg*'s more advanced features are discussed in the remaining chapters of this book.

### Step One: Setting Up the Example Application Used by This Tutorial

Before you begin this tutorial, make sure you have the example application on your system. The example application is a loan calculator called *finance*. Given a term in years, the principle loan amount and interest rate, *finance* computes the monthly payment.

The files you use for this sample application are part of the *swpkg* product and include:

- *Finance.ad*—Application defaults file.
- *Imakefile*—System independent imake description file.
- *Finance.ftr*—File typing rules file.
- *README.finance*—File describing the finance tutorial.
- *finance.man*—Manual page (nroff source).
- *main.c*—Calculator source program.

Before you begin this tutorial, set up the *finance* application by following these steps:

1. Become superuser:

```
% su
```

2. Change to the directory containing the *finance* files:

```
# cd /usr/share/src/swpkg
```

3. To create the Makefile, enter:

```
# xmkmf
```

4. To compile the *finance* application, enter:

```
# make
```

## Step Two: Running *swpkg*

To run *swpkg*, enter:

```
# swpkg
```

A small dialog box appears with a copyright notice, requesting that you wait while the system initializes. Then *swpkg* appears.

When *swpkg* first appears, it displays the first of its five main worksheets: the Create Product Hierarchy worksheet, shown in Figure 2-1.

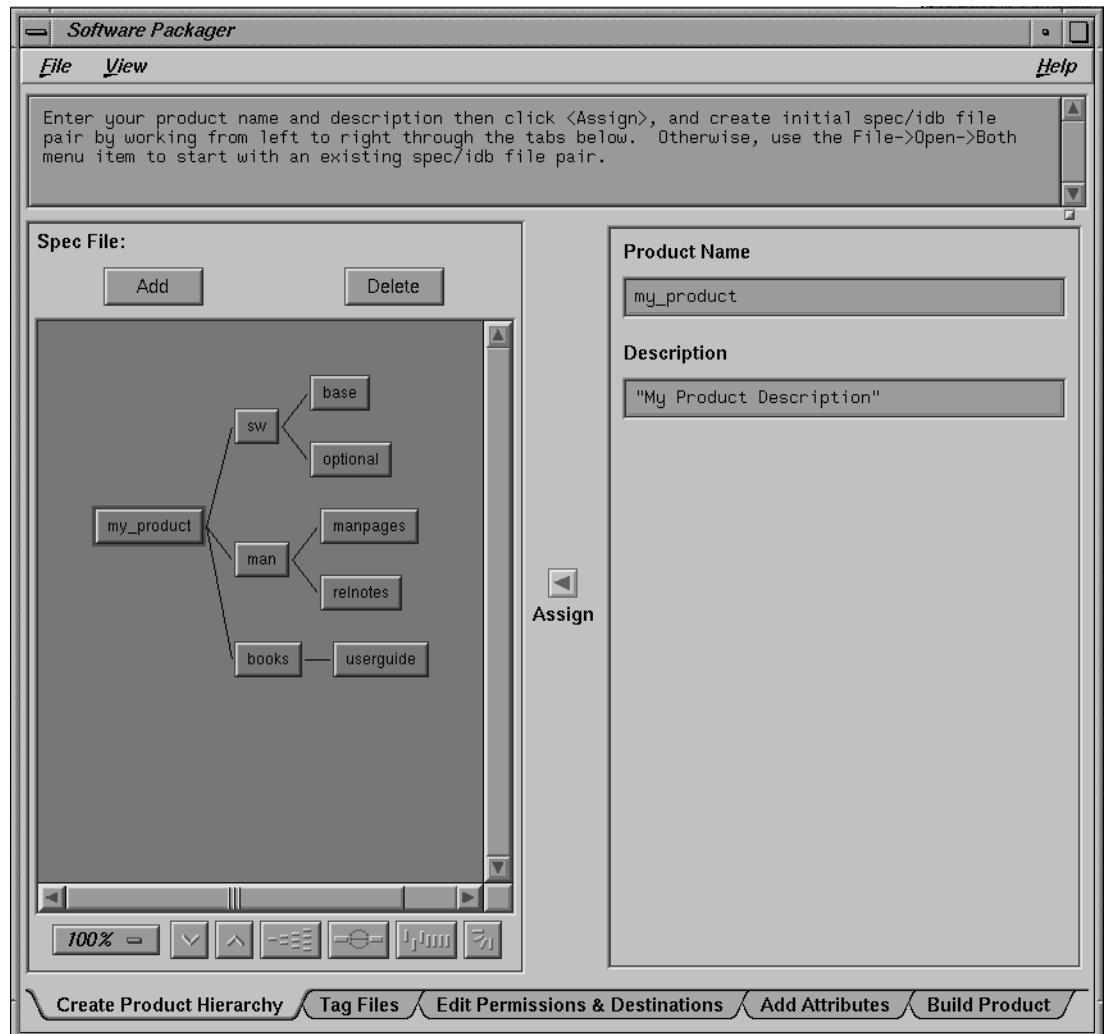


Figure 2-1 The Create Product Hierarchy Worksheet

## Step Three: Creating a Product Hierarchy

The Create Product Hierarchy worksheet helps you structure your product into three levels: product, images, and subsystems. This is called creating a *product hierarchy*. *swpkg* stores this product structure information in the *spec* file. (If you want more information now about spec files and product hierarchies, see “What’s a Spec File?” on page 40, and “About Product Hierarchies” on page 38.)

For our example program, *finance*, we’re just going to use the simplest product hierarchy: a software image and a reference page image with one subsystem each. We enter this information into the Create Product Hierarchy worksheet and tell *swpkg* to save it in the spec file.

To create the product hierarchy, follow these steps:

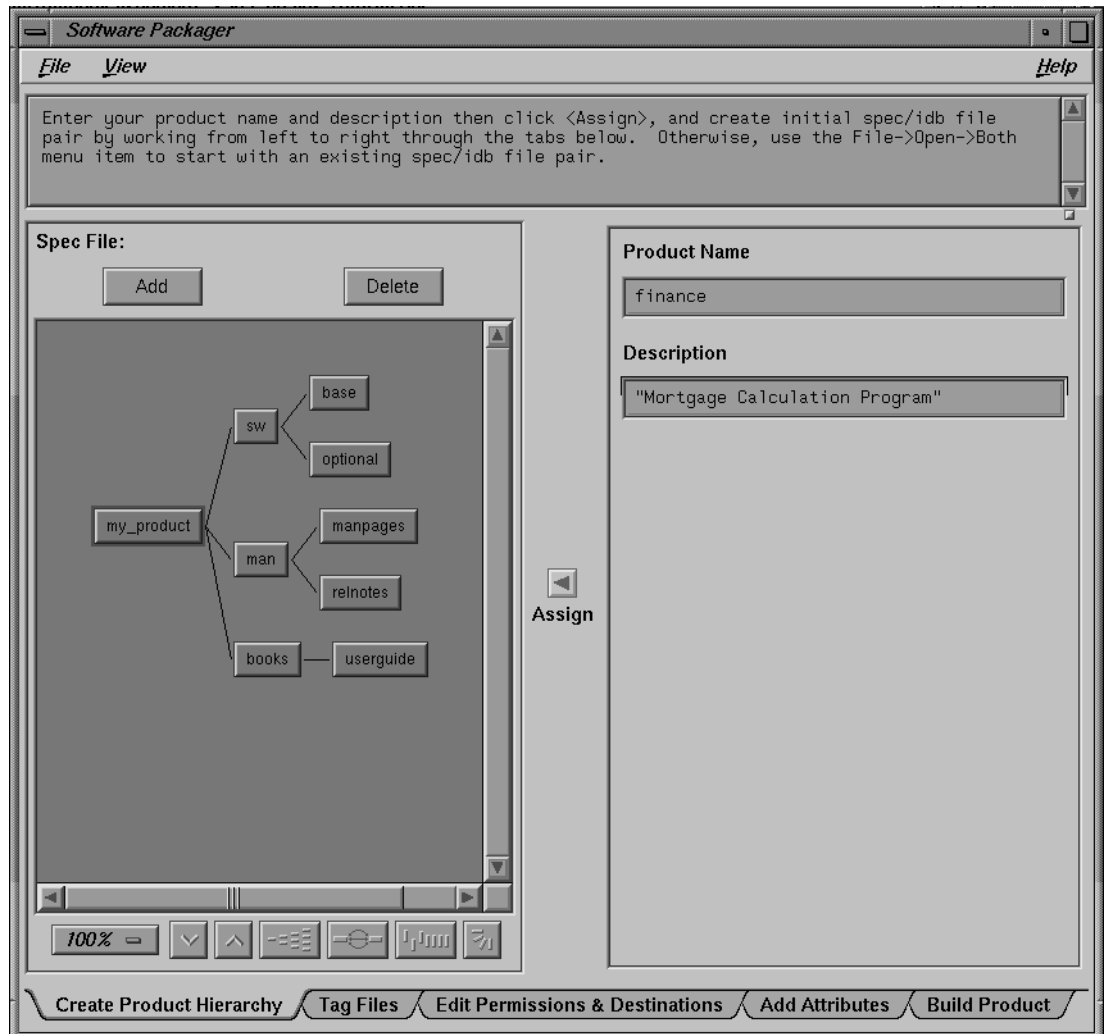
1. In the text field labeled Product Name, replace “my\_product” with “finance.” (See “Setting the Product Name” on page 48 for more detailed instructions on setting product names.)
2. In the text field labeled Product Description, replace “My Product Description” with “Mortgage Calculation Program.” (See “Setting the Product Description” on page 49 for more detailed information on setting product descriptions.)

**Note:** Remember to enclose the description in quotes. You can use either single or double quotes.

3. Click the *Assign* arrow button.

If you neglect to click the *Assign* arrow button, *swpkg* ignores your changes. After you click the *Assign* arrow button, the name of the node originally labeled “my\_product” reads “finance.”

That’s it. You’ve created a product structure for your product. Your completed worksheet should look like the one shown in Figure 2-2.



**Figure 2-2** The Completed Product Hierarchy Worksheet

**Note:** The template in the Product Hierarchy graph follows Silicon Graphics conventions for structuring and naming a product. Silicon Graphics strongly recommends that you follow these conventions. Users are presented with

this structure when they use Software Manager. The descriptions you provide for your product, images, and subsystems are the first information about your product that users see.

## Step Four: Tagging the Files

The next step is tagging the files, so that *swpkg* knows which files belong in which subsystem. To keep track of this information, *swpkg* stores this information in an installation database file (IDB file).

*swpkg* uses the IDB file to keep track of all sorts of information about the files that comprise your product, such as which files are included in which subsystem, where the files are located, and where Software Manager should put the files when it installs them. (IDB files can include other information too—for a more detailed explanation of IDB files, see “What’s an IDB File?” on page 69). For now, we’ll use the Tag Files worksheet to create a simplest-case IDB file.

First, switch to the Tag Files worksheet by clicking the tab labeled “Tag Files” (the second of the five Worksheet Selection Tabs at the top of the worksheet). When you do this, the Create Product Hierarchy worksheet is replaced by the Tag Files worksheet, shown in Figure 2-3.



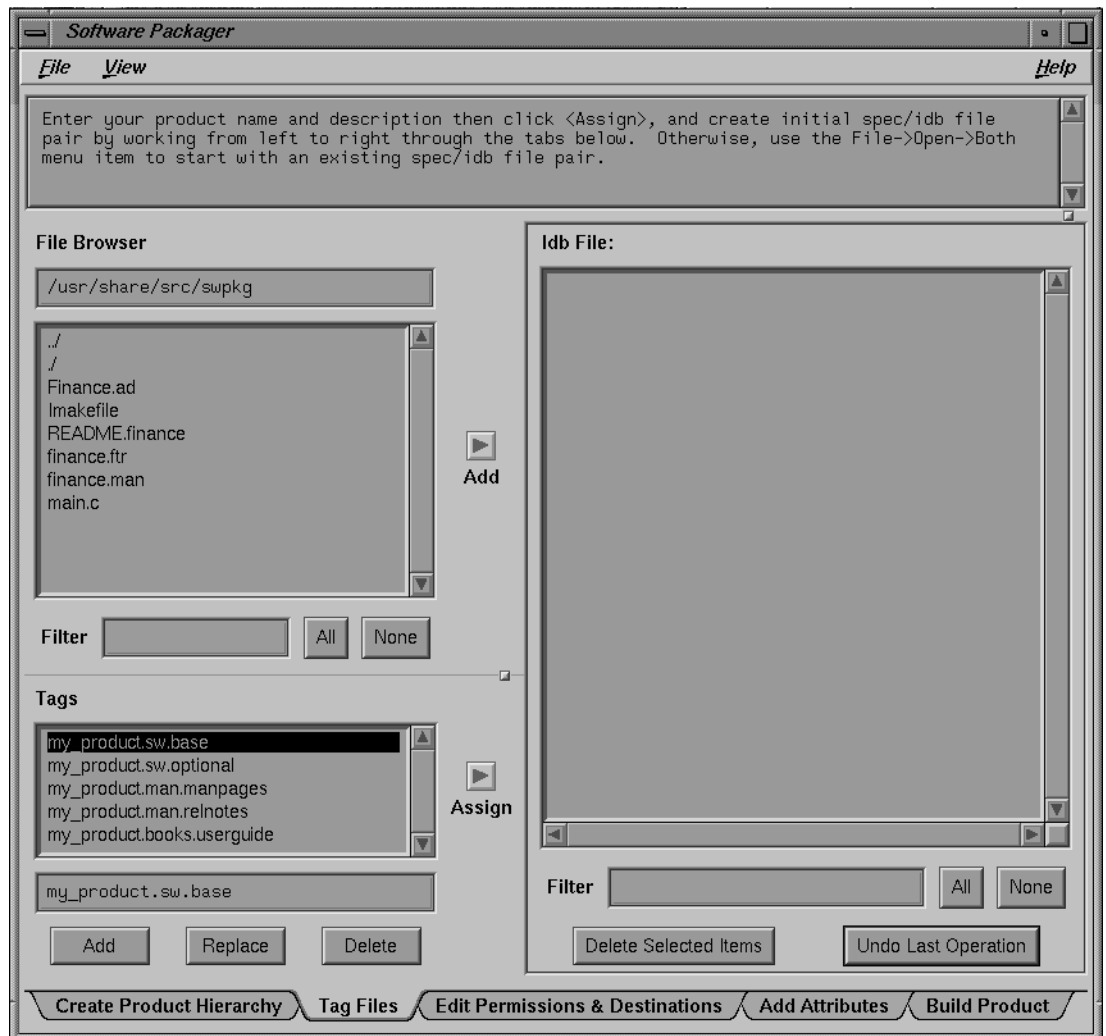


Figure 2-3 The Tag Files Worksheet

To create a simple IDB file for the Finance product, follow these steps:

1. In the field labeled File Browser, replace the text in the text:  
`/usr/share/src/swpkg`
2. Press the **<Enter>** key. The contents of the `/usr/share/src/swpkg` directory appear in the File Browser.
3. Select the files named `finance`, `Finance.ad`, `finance.ftr`, and `finance.man`.

- Using the scroll bar, scroll through the list of files until you find the `finance`.
- Click the left mouse button on `finance`.  
The line listing `finance` is highlighted in black.
- Hold down the **<Ctrl>** key and click the left mouse button on `Finance.ad`, `finance.ftr`, and `finance.man`.

4. Click the *Add* arrow button (located just to the right of the File Browser). These lines appear in the IDB File Viewer:

```
finance.sw.base usr/share/src/swpkg/Finance.ad
finance.sw.base usr/share/src/swpkg/finance
finance.sw.base usr/share/src/swpkg/finance.ftr
finance.sw.base usr/share/src/swpkg/finance.man
```

(Don't worry about the lack of the slash before the `usr` directory. `swpkg` strips the source root from the pathname and uses the root that you specify in the Source Tree Root text field in the Edit Permissions & Destinations worksheet. The default source root is slash [/].)

Notice that the lines listing the `finance`, `Finance.ad`, `finance.ftr`, and `finance.man` files now appear in bold in the File Browser. Files that are listed in the IDB File Viewer appear in bold in the File Browser so that they're easier to spot.

Notice also that `swpkg` has assigned all four of these files to the `finance.sw.base` subsystem. This is because, when we added the files to the IDB File Viewer, the tag `finance.sw.base` was selected in the Tags Browser. The `finance`, `finance.ftr`, and `Finance.ad` files do belong in the `finance.sw.base` subsystem, but we'll need to switch the `finance.man` file to the `finance.man.manpages` subsystem.

5. Click the left mouse button on the entry for `finance.man` in the IDB files list.

6. In the Tags Browser, click the left mouse button on the *finance.man.manpages* tag. The line listing *finance.man.manpages* is highlighted in black.
7. Click the left mouse button on the *Assign* arrow button.

The IDB File Viewer now shows this line for the *finance.man* file:

```
finance.man.manpages usr/share/src/swpkg/finance.man
```

Now the *finance.man* file is assigned to the *finance.man.manpages* subsystem.

Now *swpkg* knows which files belong in which subsystems. Your completed Tag Files worksheet should look like the one shown in Figure 2-4.

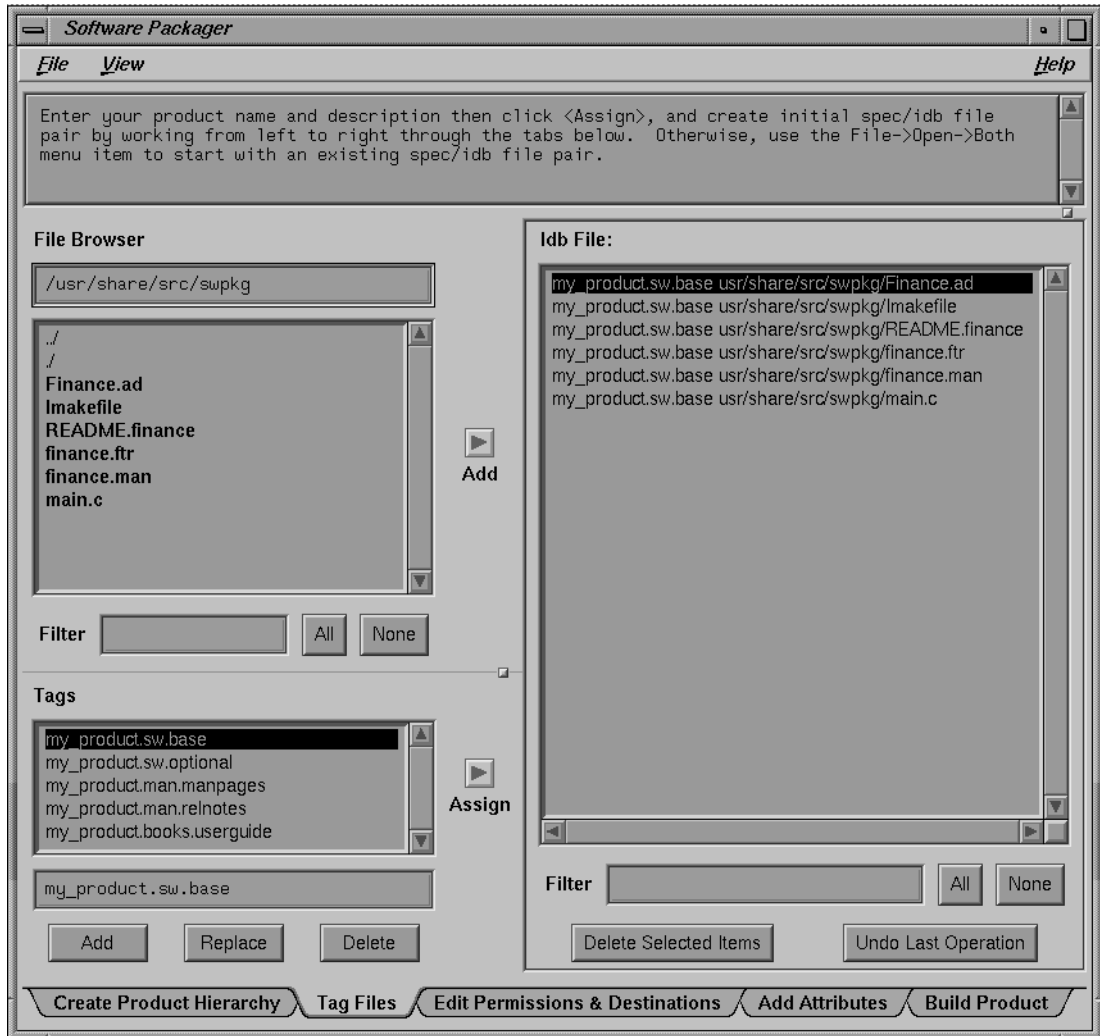
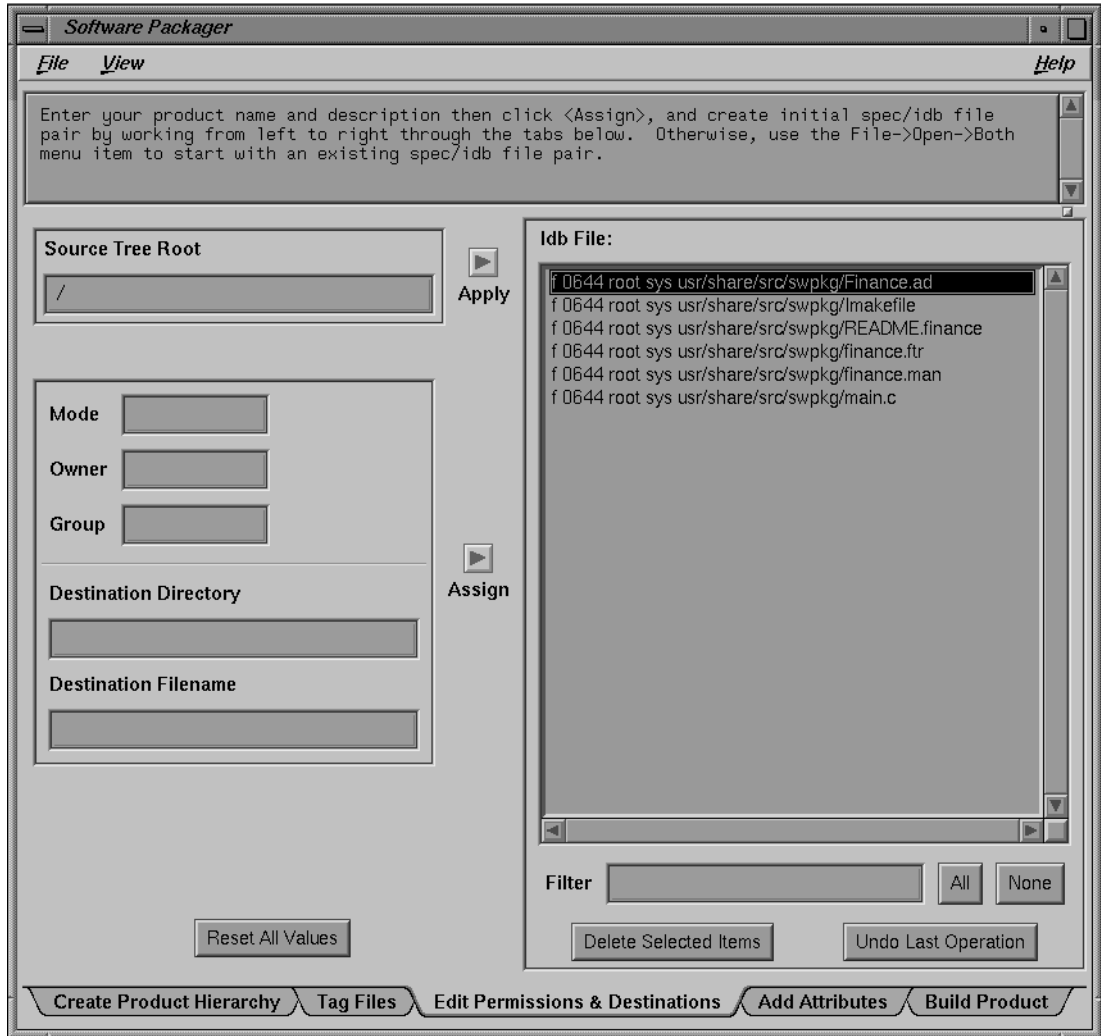


Figure 2-4 The Completed Tag Files Worksheet

## Step Five: Setting Permissions and Destinations

*swpkg* now knows which files are in which subsystems and where to find each file, but it doesn't know where you want the files installed or what the permissions should be. You provide this information using the Edit Permissions & Destinations worksheet, and *swpkg* stores it in the IDB file.

Open the Edit Permissions & Destinations worksheet by clicking the worksheet selection tab labeled "Edit Permissions & Destinations." The Tag Files worksheet is replaced by the Edit Permissions & Destinations worksheet, shown in Figure 2-5.



**Figure 2-5** The Edit Permissions & Destinations Worksheet

In the Edit Permissions & Destinations worksheet, you enter permissions and destination information for each file in your product. *swpkg* stores the information you enter in this worksheet in the IDB file. Later, when users install your product, Software Manager uses these lines to figure out where to install the files and how to set the file modes, owners, and groups.

To edit the permissions information for the *Finance.ad* file, follow these steps:

1. In the IDB File Viewer, double-click on the *Finance.ad* file. The current values for all the fields in the worksheet appear.
2. In the Mode text field replace the value with:  
**444**
3. In the Owner text field replace the value with:  
**root**
4. In the Owner text field replace the value with:  
**sys**
5. In the Destination Directory text field replace the value with:  
**usr/lib/X11/app-defaults**
6. In the Destination Filename text field replace the value with:  
**Finance**
7. Click the *Assign* arrow button.

We're going to leave the Source Tree Root text field alone, since we want to use the default value (/). For information on setting source roots, see "Setting Source Tree Roots" on page 86.

You've finished filling in the worksheet for the *Finance.ad* file. The Edit Permissions & Destinations worksheet should now look like the one shown in Figure 2-6. For more detailed information on working with the text fields in the Permissions and Destinations sheet, see "Setting Permissions and Destination Directories" on page 88.

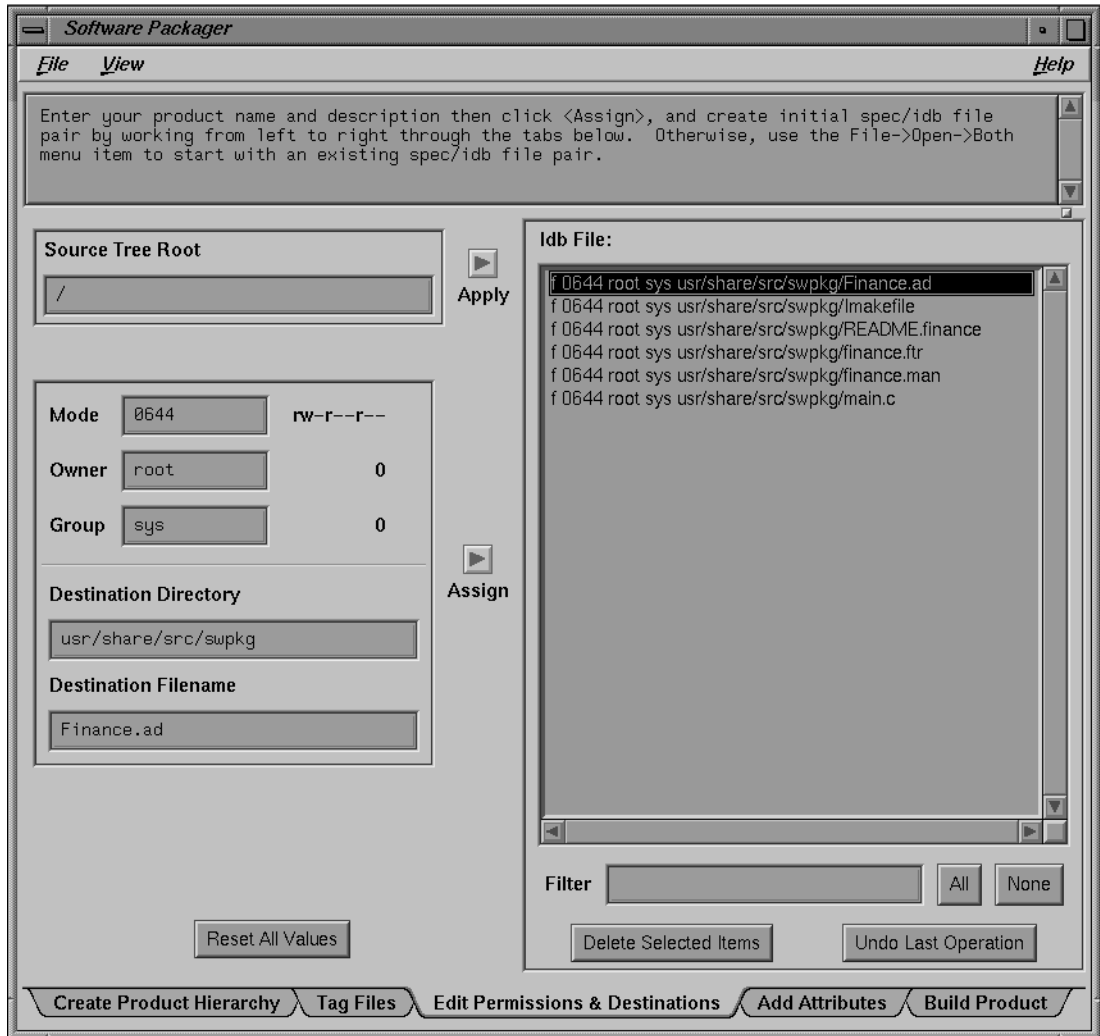


Figure 2-6 The Completed Worksheet for the *Finance.ad* File



Now, using Table 2-1 as your guide, edit the permissions and destinations information for each of the remaining files:

1. Double-click a file.
2. Complete its data as shown in Table 2-1:
3. Click the *Assign* arrow button.

**Table 2-1** Permissions and Destinations for Remaining Files

File	Mode	Owner	Group	Destination Directory	Destination Filename
<i>finance</i>	775	root	sys	usr/bin/X11	finance
<i>finance.ftr</i>	444	root	sys	usr/lib/filetype/install	finance.ftr
<i>finance.man</i>	644	root	sys	usr/man/u_man/man1	finance.1

You've finished filling in the Edit Permissions & Destinations worksheet, now you're ready to add installation attributes to your product.

## Step Six: Adding Attributes

The next step is to add any necessary installation attributes to the product's files. To add an installation attribute, you need to switch from the Edit Permissions & Destinations worksheet to the Add Attributes worksheet. Click the left mouse button on the Worksheet Selection Tab labeled "Add Attributes." The Add Attributes worksheet, shown in Figure 2-7, appears.

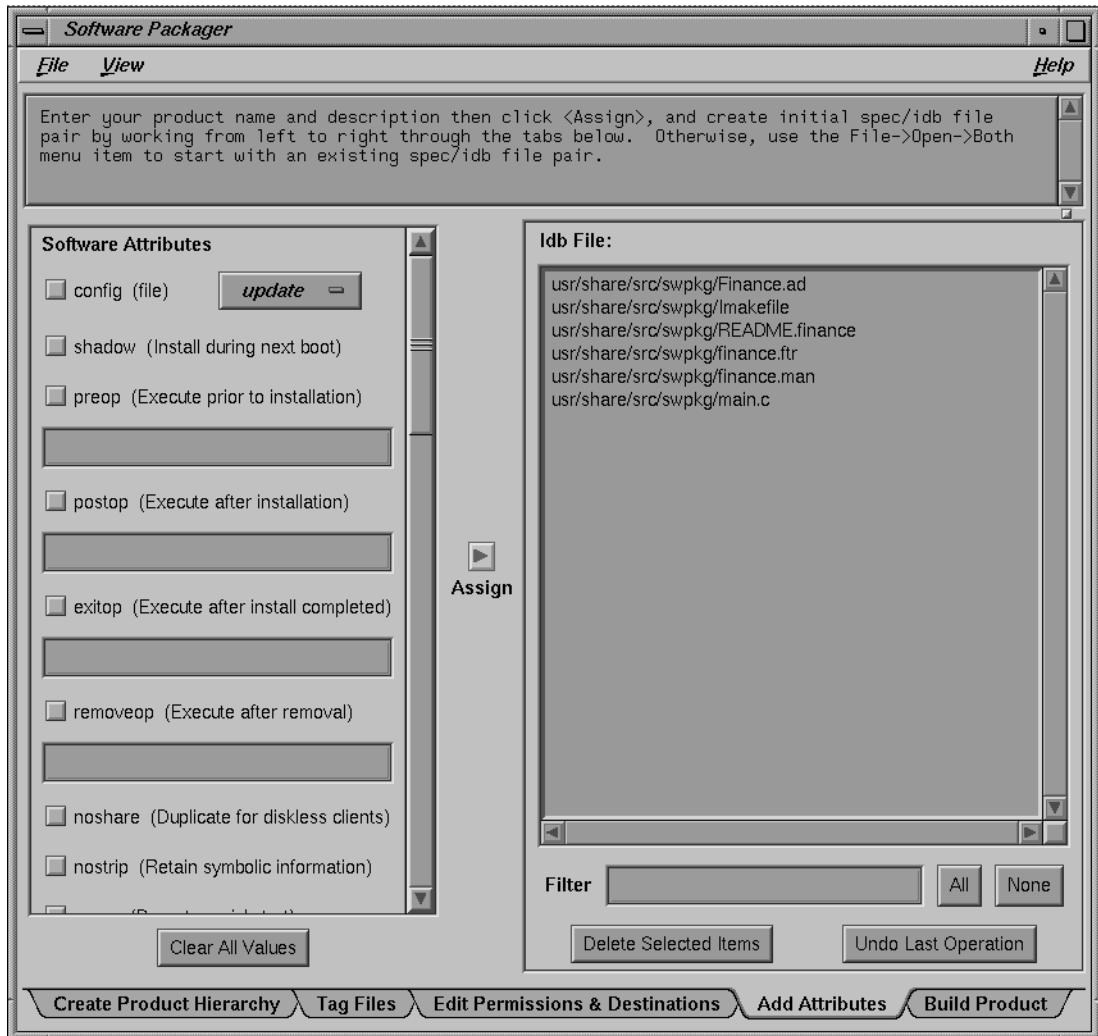


Figure 2-7 The Add Attributes Worksheet

This worksheet allows you to set up certain installation attributes for the files in your product. In this example, we'll go with the simplest case: we'll use the exitop installation attribute to tell Software Manager to install the application's icon in the Icon Catalog after the user gives the quit command.

1. Double-click the left mouse button on the first item in the IDB File Viewer, the *Finance* executable.
2. Click the left mouse button on the exitop check button under "Software Attributes."
3. In the text field under the exitop check button, type:

```
"tag 0x000101A0 /usr/bin/X11/finance  
iconbookedit -add 'Category:File Name:/usr/bin/X11/finance' -syspage DesktopTools  
cd /usr/lib/filetype  
make -u"
```

**Note:** The entire list of commands is enclosed in a pair of double-quotes (" ").

4. Click the *Assign* arrow button.

For more information on the *iconbookedit* command, see section "Step Five: Installing Your Application in the Icon Catalog," and for more information on updating the Desktop database, see section "Step Four: Compiling the Source Files." Both sections are in Chapter 11 of the *Indigo Magic Desktop Integration Guide*.

You've finished filling in the Add Attributes worksheet. The worksheet should now look like the one shown in Figure 2-8.

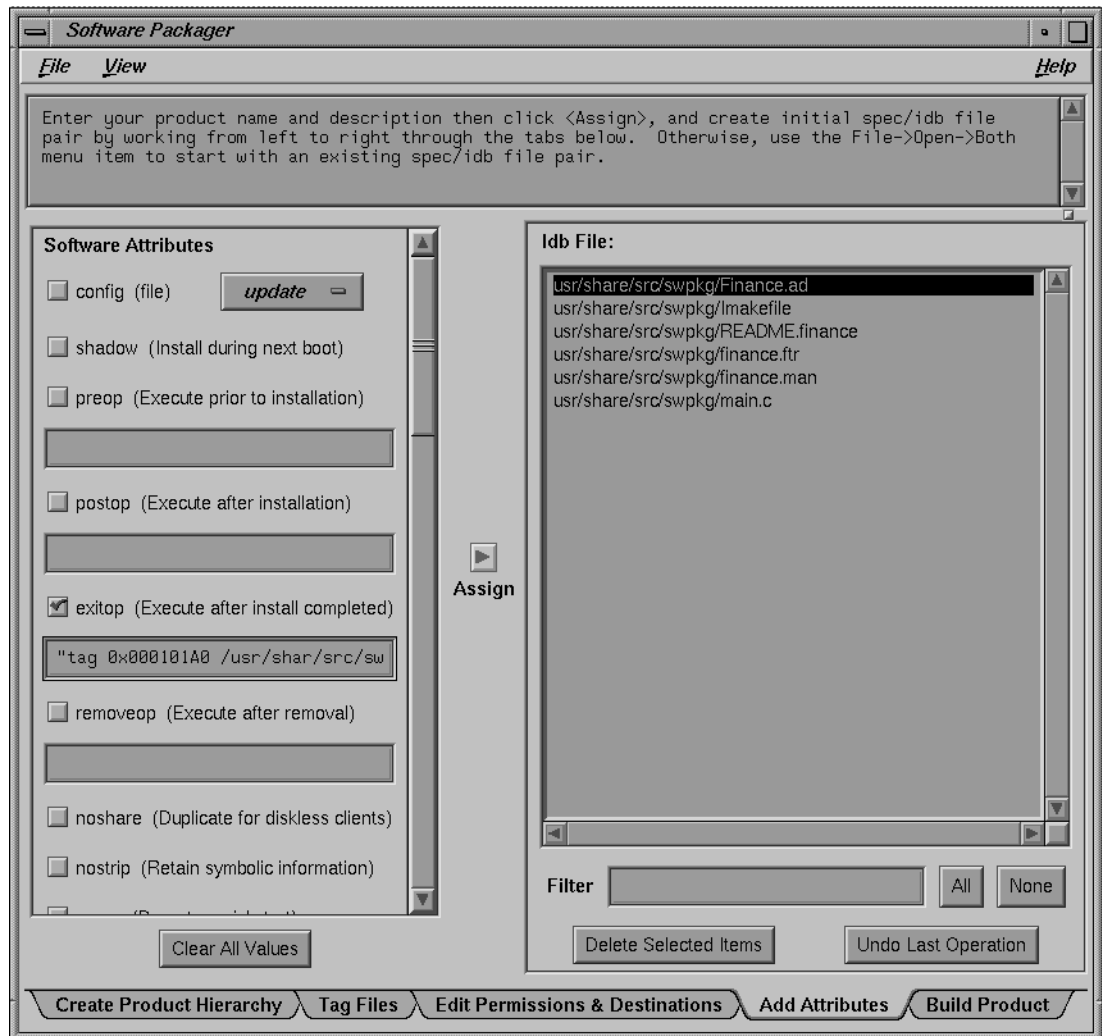


Figure 2-8 The Completed Add Attributes Worksheet

## Step Seven: Building the Product

To build your product, you need to switch to the Build Product worksheet. Click the left mouse button on the Worksheet Selection Tab labeled "Build Product." The Build Product worksheet, shown in Figure 2-9, appears.

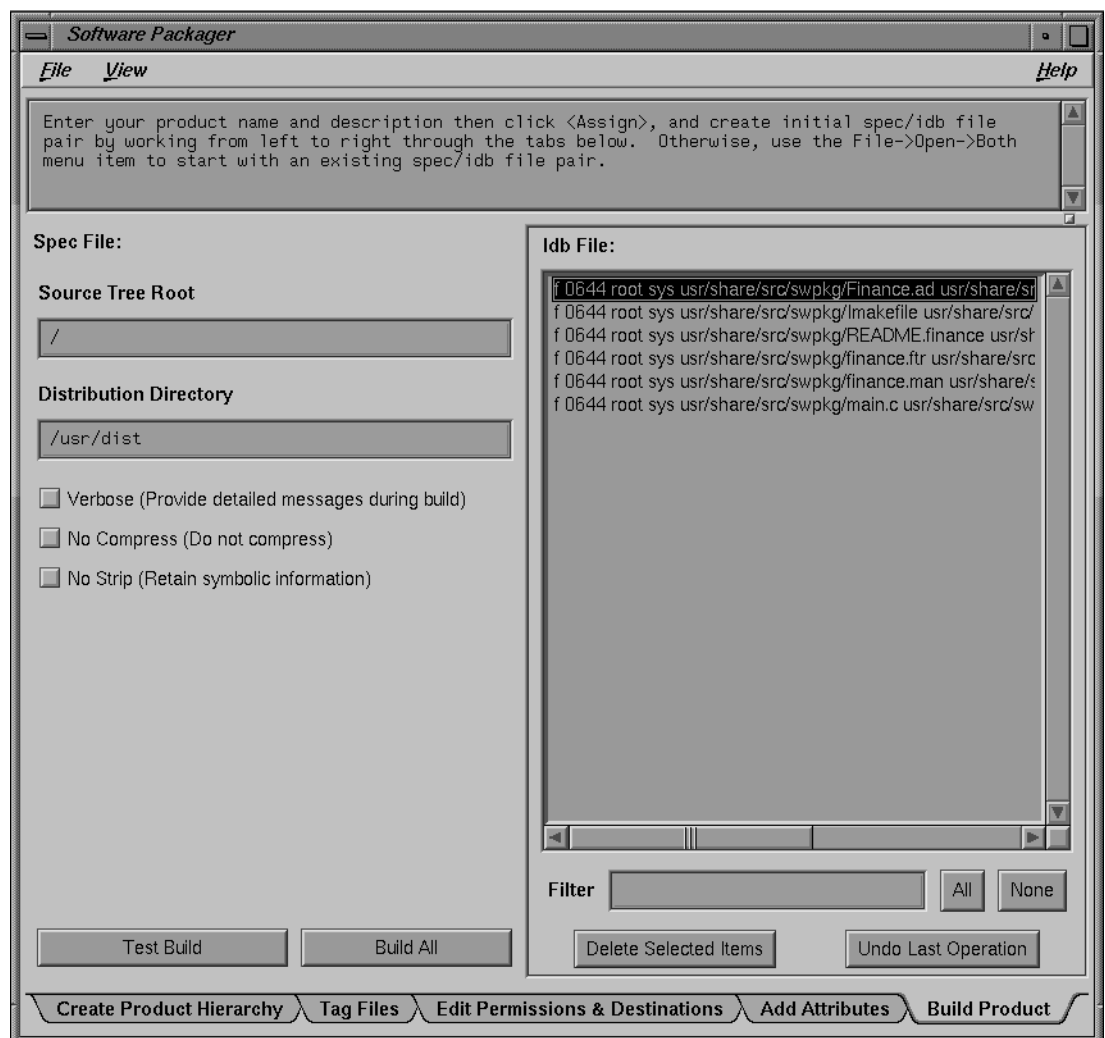


Figure 2-9 The Build Product Worksheet

This worksheet lists the lines in the IDB file and shows the settings for the source root and the distribution directory. (The distribution directory is the directory where you want *swpkg* to put all the installable files it creates. This directory is often named *dist*, which is short for distribution.)

For this simplest-case example, we're going to stick with the default distribution directory, */usr/dist* and we won't select any of the Build Options check buttons (located below the distribution directory text field). For more information on the build options, see "Selecting Build Options" on page 116.

Before building the product, it's a good idea to try a test and see if any problems crop up without having *swpkg* actually try to build the product.

1. Click the left mouse button on the button labeled *Test Build* at the bottom right corner of the worksheet.

If you haven't already saved your spec and idb files, *swpkg* displays a Save dialog asking where you want them saved.

2. Click the OK button in the Save dialog for both files.

If *swpkg* runs into problems, it lists them in the Message Area near the top of the worksheet.

If no problems are listed, you're ready to build the product. To build the product:

1. Click the left mouse button on the *Build All* button at the bottom of the worksheet.

As *swpkg* builds your product, it displays any error messages in Message Area. When it's finished, it displays a message to that effect in the Message Area. Now you're ready to quit *swpkg*.

2. Quit by selecting the "Quit" menu item from the File pull-down menu (described in "Using the File Menu" on page 5).

## **Step Eight: Installing and Running the Product**

After you've built the product, use Software Manager to install it on your local workstation. Look at the short and long names Software Manager lists for your product, images, and subsystems—are they clear and distinct? After your product is installed, quit Software Manager. Were all the files installed in the correct places? Does the icon appear in the Icon Catalog?





## **Creating a Product Hierarchy**

The first step in creating an installable product is to define the product's *hierarchy*, the structure of the subsystems within your product.



---

## Creating a Product Hierarchy

This chapter explains the first step for packaging your application for installation: creating a product hierarchy—an installation structure—for your product. This chapter contains these sections:

- “Creating a Product Hierarchy: Before You Begin” on page 37 provides some background information and lists the prerequisites for creating a product hierarchy.
- “Creating a Product Hierarchy: The Basic Steps” on page 41 lists the basic steps for creating a product hierarchy and explains where to find more detailed instructions for each step.
- “Using the Create Product Hierarchy Worksheet” on page 42 describes the features of the worksheet and explains how to use these features to create your product hierarchy.

### Creating a Product Hierarchy: Before You Begin

This section lists the prerequisites for creating a product hierarchy and defines spec files and product hierarchies.

#### Prerequisites

To package an application successfully, you need know

- what objects need to be built
- how these objects should be grouped within the product
- where the objects are located
- where they should be installed on the users’ workstations
- what permissions they should have on the users’ workstations

## About Product Hierarchies

*swpkg* requires that you organize your application's files into a three-level hierarchy. The highest level is the *product* level, the second level is the *image* level, and the third level is the *subsystem* level.

At the product level, software is grouped into distinct products. At the image level, a single product's files are grouped according to type: for example, one image for the software, and another for reference pages. At the subsystem level, the files in each image are organized into groups of files that are installed as a unit.

The purpose is to create subsystems consisting of related files that your users might want to install (or decline to install) as a group. For example, if your application provides several optional templates, you might group them together into a single subsystem. That way, users who are short on disk space, or who don't think they need the optional templates, can choose not to install that particular subsystem. Similarly, if all your documentation files (reference pages, release notes, online help, online books, and so on) are grouped into a single image, users can choose to install them all together, rather than subsystem by subsystem. So, if you choose your product structure carefully, you can make the installation process much easier for your users.

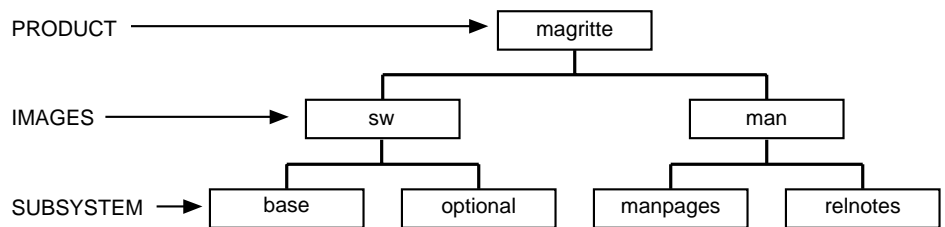
## An Example Product Hierarchy

Here's an example of how you might create a product hierarchy for a simple application. Suppose you want to package a paint application called *magritte*. Let's say that the *magritte* application includes the basic application software, some clip art, a reference page, and some release notes. Here's a good way to structure your files for installation:

1. Separate the software files (the application software and the clip art) and the documentation (the reference page and release notes) into two separate images, called *sw* and *man*, respectively.
2. Divide the software image (*sw*) into two subsystems: one, called *base*, containing the base application software; and the other, called *optional*, containing the clip art.

3. Divide your documentation image (*man*) into two subsystems: one, called *manpages*, containing the reference page and the other, called *relnotes*, containing the release notes.

In this example, our product hierarchy would be one product (*magritte*) containing two images (*sw* and *man*), with each image containing two subsystems (*base* and *optional*, and *manpages* and *relnotes*, respectively). Figure 3-1 illustrates the structure of this example product hierarchy.



**Figure 3-1** Example Product Hierarchy

### Naming Images and Subsystems

Image and subsystem names must reflect the product hierarchy by following these naming conventions:

- Each image has a name of the form:

`product.image`

For example, if the name of the product is *magritte*, and the image is *sw*, then the full image name would be:

`magritte.sw`

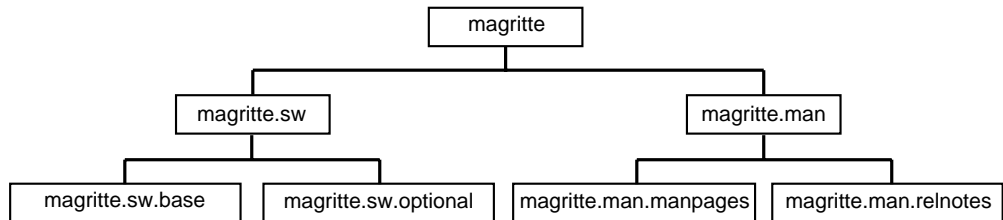
- Each subsystem has a name of the form:

`product.image.subsystem`

For example, if the name of the product is *magritte*, the image is *sw*, and the subsystem is *base*, then the full subsystem name would be:

`magritte.sw.base`

Figure 3-2 shows the correct product, image, and subsystem names, for the example introduced in “About Product Hierarchies” on page 38.



**Figure 3-2** Example Product Hierarchy Names

### Silicon Graphics Conventions for Product Hierarchy

This section lists the Silicon Graphics conventions for structuring your product hierarchy.

Silicon Graphics conventions strongly recommend that:

- Your product must contain an image named *sw* that contains all the software subsystems for your product.
- The *sw* image must contain a subsystem named *base* that contains the base software for your product. (You might put optional software in a subsystem named *optional*, but this is not required.)
- If your product contains reference pages or release notes, it must also include an image named *man* that contains the subsystems that include the reference pages and release notes.
- The subsystem containing the reference pages must be named *manpages* and the subsystem containing the release notes must be named *relnotes*.

You are not required to follow these conventions, but Silicon Graphics strongly recommends that you do.

### What's a Spec File?

*swpkg* stores information about the product hierarchy in the product specification (spec) file for your product. When you enter information about your product hierarchy in the Create Product Hierarchy worksheet, you are actually creating a spec file for your product.

In general, here's what goes in a spec file:

- product, image, and subsystem names—these names appear in Software Manager and *versions* listings
- product, image, and subsystem descriptions—these descriptions appear in Software Manager and *versions* listings
- installation order
- initial installation information
- installation prerequisites and incompatibilities
- automatic removal of obsolete subsystems
- the version number

## Creating a Product Hierarchy: The Basic Steps

These are the steps for creating a product hierarchy:

1. Edit the nodes in the template displayed in the Product Hierarchy graph so that the graph has the right number of images, each containing the right number of subsystems.

If the template in the Product Hierarchy graph does not provide the right number of images and/or subsystems for your product hierarchy, add or delete existing images or subsystems as necessary. (See "Adding and Deleting Nodes" on page 45 for instructions.)

2. Edit the Product Specification sheet to set the product name and description. (See "Entering Product Specifications" on page 47 for instructions.)
3. Edit the Image Specification sheet to set image names and descriptions and, if necessary, to specify any of these installation options:
  - installation order
  - version number (instead of using the default value)(See "Entering Image Specifications" on page 50 for instructions.)

4. Edit the Subsystem Specification sheet to set subsystem names and descriptions and, if necessary, to specify any of these installation options:

- default installation
- reboot installation

You might also need to specify:

- prerequisite subsystems
- incompatible subsystems
- replacement subsystems

(See “Entering Subsystem Specifications” on page 55 for instructions.)

5. Save the spec file by selecting “Spec” from the “Save” rollover menu in *swpkg*’s File menu.

## Using the Create Product Hierarchy Worksheet

When *swpkg* first appears, it displays the Create Product Hierarchy worksheet, shown in Figure 3-3. This section describes the features of the worksheet and explains how to use it to create a product hierarchy.



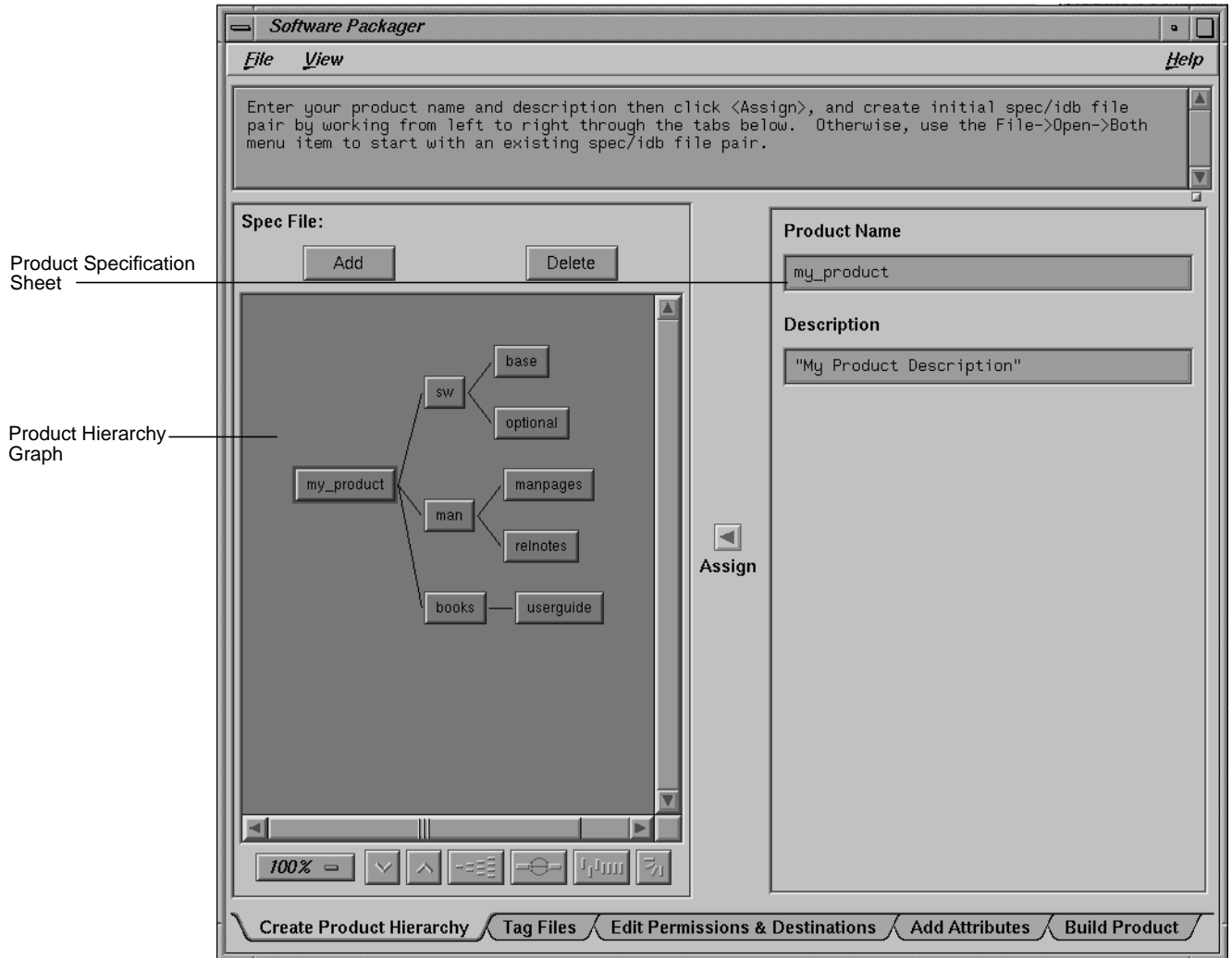
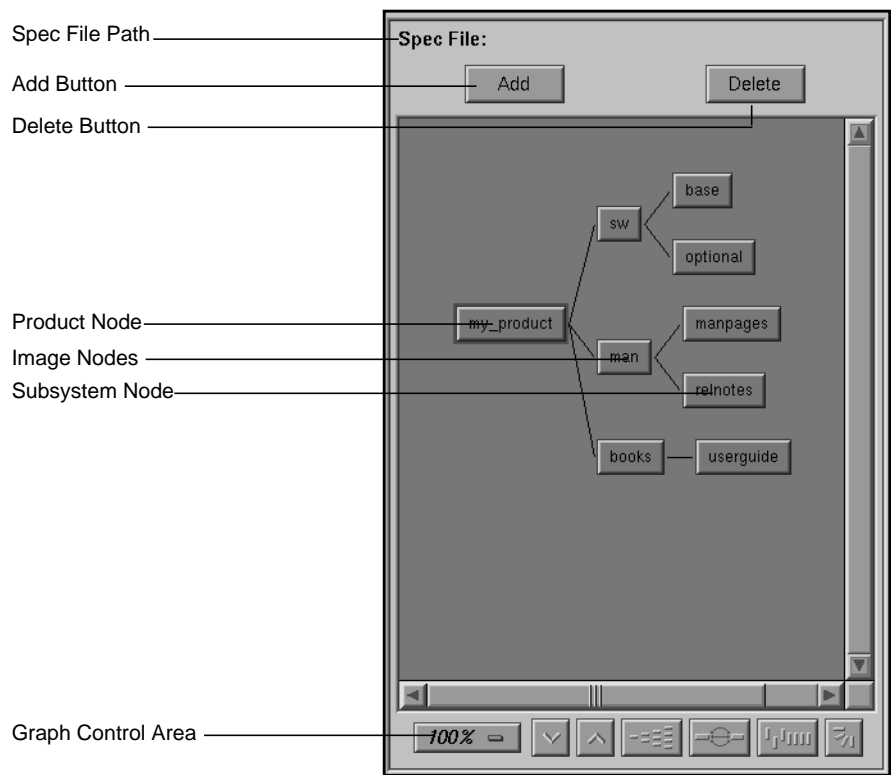


Figure 3-3 The Create Product Hierarchy Worksheet

### Creating a Product Structure

The Product Hierarchy graph, shown in Figure 3-4, displays a template for a standard product hierarchy. You can alter this template to reflect a new product hierarchy. If you open an existing spec file, the Product Hierarchy graph displays the product hierarchy for that spec file. (To open an existing spec file, use the “Open” item in the File menu.)



**Figure 3-4** The Product Hierarchy Graph

Use the Product Hierarchy graph to create a structure for your product hierarchy—that is, arrange the nodes in the graph so that the graph has the right number of images, each containing the right number of subsystems. (For an explanation of nodes, see “Selecting Nodes” on page 45.) Add or delete nodes as necessary (see “Adding and Deleting Nodes” on page 45 for instructions).

For a general discussion of product hierarchies, along with guidelines on structuring your product, refer to “About Product Hierarchies” on page 38.

### **Selecting Nodes**

The Product Hierarchy graph provides a graphical display of your product hierarchy, in which products, images, and subsystems are shown as rectangles (or *nodes*) and relationships as connecting lines (or *arcs*). (If the graph is larger than the viewing area, scroll bars are enabled.)

When you select a node, the Specification sheet associated with that node appears to the right of the Product Hierarchy Graph. (The Specification sheets are groups of text fields and/or check buttons that you can use to provide the relevant installation information about each product, image, and subsystem in your installation plan.)

To select a node, click it with the left mouse button. The Specification sheet for that node appears to the right of the Product Hierarchy Graph. The Create Product Hierarchy worksheet has three different types of Specification sheets: the Product Specification sheet (described in “Entering Product Specifications”), the Image Specification sheet (described in “Entering Image Specifications”), and the Subsystem Specification sheet (described in “Entering Subsystem Specifications”).

### **The Spec File Path Label**

The spec file path label shows the current spec file pathname. If no valid spec file is identified, the path is left blank. For information on creating a spec file or changing the spec file path, see “Using the File Menu” on page 5. For a definition of a spec file, read “What’s a Spec File?” on page 40.

### **Adding and Deleting Nodes**

Use the *Add* button (located at the top of the product hierarchy graph) to add products, images, and subsystems to your product hierarchy (spec file):

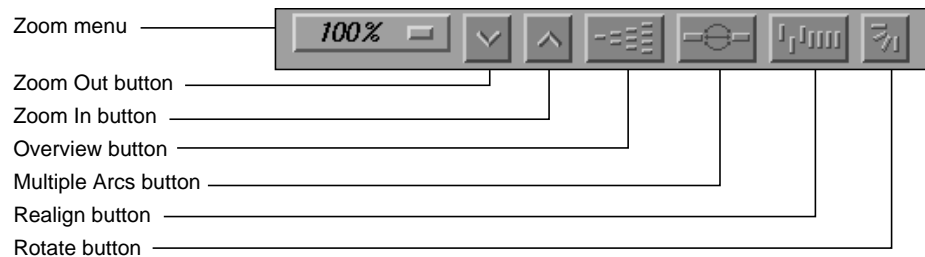
- To add a new product, make sure no nodes are selected (by clicking the background of the Product Hierarchy Graph), then click the *Add* button.

- To add a new image, select the node for the product to which the image will belong, then click the *Add* button.
- To add a new subsystem, select the node for the image to which the subsystem will belong, then click the *Add* button.

Use the *Delete* button to delete a product, image or subsystem by selecting the relevant node and clicking the *Delete* button.

### Selecting Display Options for the Product Hierarchy Graph

The product hierarchy graph has a control area containing a row of graph controls, shown in Figure 3-5.



**Figure 3-5** Graph Display Controls

These graphical view controls are:

#### Zoom menu

Shows the current scale of the graph. If clicked, a popup menu appears displaying other available scales. The scaling range is between 15% and 300% of the normal (100%) size.

#### Zoom Out button

Resets the scale of the graph to the next available smaller size in the range.

#### Zoom In button

Resets the scale of the graph to the next available larger size in the range.

**Note:** If you drag a node into a new position and then use one of the *Zoom* buttons, the node returns to its initial position.

*Overview* button

Displays the Overview window, which lets you view the entire graph at a reduced scale. The Overview window has a movable viewport that lets you select the portion of the graph displayed in the main window. The Overview window also has an Admin menu with these three selections:

- “Scale to Fit” scales the graph to match the aspect ratio of the overview window.
- “Show Arcs” displays or hides the arcs between the nodes.
- “Close” closes the Overview window.

*Multiple Arcs* button

Toggles between single and multiple arc mode. (This button is not useful in the Product Hierarchy Graph.)

*Realign* button Redraws the graph, restoring the positions of any nodes that were repositioned.

*Rotate* button Flips the orientation of the graph between horizontal (calling nodes at the left) and vertical (calling nodes at the top).

**Note:** If you reposition the nodes by dragging and then change orientation, the nodes will return to their initial positioning relative to each other.

## Entering Product Specifications

Use the Product Specification sheet to enter your product’s name and description. The Product Specification sheet, shown in Figure 3-6, is visible whenever a product node is selected in the Product Hierarchy graph. For example, if you click the left mouse button on the node labeled *my\_product* in the product hierarchy graph, the Product Specification sheet for the *my\_product* product appears to the right of the product hierarchy graph.



**Figure 3-6** The Product Specification Sheet

To enter information in the text fields, just delete the existing text, type in the new text, and click the *Assign* arrow button. **If you don't click the *Assign* arrow button, your changes will be lost when you select a different node.**

### Setting the Product Name

To name the product, follow these steps:

1. Delete the existing text in the Product Name text field.
2. Type in the new text.
3. Click the *Assign* arrow button.

The product name is the short name for the product—this is the “short” name that appears in Software Manager listings (the “long” name is the

product description). Because the display length for both long and short names in Software Manager is typically 30 characters, the product name should be short so that subsystem names are less likely to overflow their column in *versions* and Software Manager listings. Short product names are also easier for users to enter. Do not choose a product name that begins with a digit.

### Setting the Product Description

To change the product description, follow these steps:

1. Delete the existing text in the Product Description text field.
2. Type in the new description.

**Note:** Always enclose your description within quotes. You can use either single or double quotes. You can use one type of quotes as part of your description if you use the other type of quotes to quote the description. For example, 'Acme 1/2" Tape Support' is a valid description as long as you enclose it in single quotes.

3. Click the *Assign* arrow button.

The product description is a brief description of your product that begins with the product name. This description is the "long" name that appears in Software Manager listings (the description also appears in *versions* listings). The description should be brief but informative, since this might be the only information users have to help them to decide whether to install your software.

In particular, the product description should begin with the marketing name of the product. Do not use the words "option," "version," "system," or "release," unless they are part of the name of the product (for example, Maintenance Release and Network File System).

Here are some general guidelines for creating descriptions:

- Limit the string to 30 characters, if possible. Anything larger gets wrapped or truncated by Software Manager and *versions*.
- When abbreviating words, use only standard, easily understood abbreviations.
- Capitalize the first letter of each word, except prepositions.

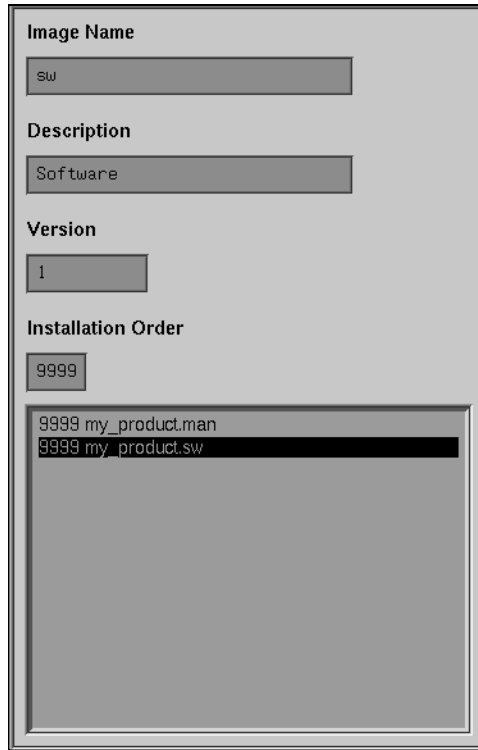
- Do not use punctuation.
- Do not include part numbers or marketing codes
- Include a market version number (version 1.2, for example)

So, for example, assume your product is a mortgage calculation application called Mortgage Calculator, that this is the first version of the product, and that its marketing code is M3-486. A good product description would be: "Mortgage Calculator 1.0"—a bad one would be: "Mort.Cal.M3-486."

### Entering Image Specifications

The Image Specification sheet, shown in Figure 3-7, is visible whenever an image node is selected in the Product Hierarchy graph. For example, if you click the left mouse button on the image labeled *sw* in the Product Hierarchy Graph, the Image Specification sheet for the *sw* image appears to the right of the Product Hierarchy Graph.





The screenshot shows a form titled "Image Specification Sheet" with several text input fields and a list box. The fields are labeled "Image Name", "Description", "Version", and "Installation Order". The list box contains two entries: "9999 my\_product.man" and "9999 my\_product.sw".

<b>Image Name</b>	sw
<b>Description</b>	Software
<b>Version</b>	1
<b>Installation Order</b>	9999
<b>Image List</b>	9999 my_product.man 9999 my_product.sw

**Figure 3-7** The Image Specification Sheet

You can select any image in this Specification sheet by clicking the image name in the list of images below the text fields.

Use the Image Specification sheet to enter the image's name, description, version number, and installation order (each of these text fields is discussed in detail in this section).

To enter information in the text fields, delete the existing text, type in the new text, and click the *Assign* arrow button. **If you don't click the *Assign* arrow button, your changes will be lost when you select a different node.**

### Naming the Image

To change the name of an image, follow these steps:

1. Select the appropriate image node in the Product Hierarchy graph.
2. Delete the existing text in the Image Name text field.
3. Type in the new name for the image.
4. Click the *Assign* arrow button.

Refer to “Naming Images and Subsystems” on page 39 for guidelines.

### Changing the Description of the Image

To change the description of an image, follow these steps:

1. Select the appropriate image node in the Product Hierarchy graph.
2. Delete the existing text in the Description text field.
3. Type in the new description for the image.
4. Click the *Assign* arrow button.

**Note:** Always enclose your description within quotes. You can use either single or double quotes. You can use one type of quotes as part of your description if you use the other type of quotes to quote the description. For example, ‘Acme 1/2” Tape Support’ is a valid description as long as you enclose it in single quotes.

If the image in question is the *sw* or *man* image, you can just leave the description as it is in the corresponding (*sw* or *man*) template.

The image description is a brief description of the image. Since this description appears in Software Manager listings, it’s important that the description be informative and distinct from the product and subsystem descriptions.

Like the product description, the image description should begin with the marketing name of the product. This is because Software Manager lists the products, images, and subsystem according to these descriptions—if your product, image, and subsystem names begin differently, they aren’t listed together.

Image descriptions should include the product name and a description of the image (for example, Software, Documentation, or Manual Pages). Do not include a version or release number.

Here are some general guidelines for creating descriptions:

- Limit the string to 30 characters, if possible. Anything larger gets wrapped or truncated by Software Manager and *versions*.
- When abbreviating words, use only standard, easily understood abbreviations.
- Capitalize the first letter of each word, except prepositions.
- Do not use punctuation.
- Do not include part numbers or marketing codes.

### **Changing the Version Number of the Image**

Choose a version number for your images. You can keep the existing, default value for the version number or you can create a version number of your own. Some rules and suggestions for selecting version numbers are:

- The version number can contain no more than ten digits.
- The version number cannot contain a decimal.
- Do not use your software release number as the version number.
- Start with a low version number. The version number of each software distribution you create for your product is higher than the last value. Over the lifetime of your product, 999999999 is the highest version number you can use. After that, you must change product or image names.

Type the version number directly into the text field labeled “Version,” then click the *Assign* arrow button.

### **Changing the Installation Order of the Image**

The number in the text field labeled Installation Order tells Software Manager the order in which to install the various images. If you want to specify that some of your images be installed before others, select the

installation order numbers accordingly. (Remember to click the *Assign* arrow button after typing in each order number.)

Installation order is specified by numbers in the range of 0 to 9999. The lower the order number, the earlier that image will be installed. If you do not specify an order number, the default is 9999 (the highest order, meaning the last to be installed). Images with equal order numbers are installed in alphabetical order.

Software Manager installs products in alphabetical order by default. Within each product, images are also installed in alphabetical order. Similarly, the subsystems in each image are installed in alphabetical order. For example, suppose there are two products, *a* and *b*. Each of them contains two images, *sw* and *man*. Each image has two subsystems, *x* and *y*. The installation order of these subsystems is:

```
a.man.x
a.man.y
a.sw.x
a.sw.y
b.man.x
b.man.y
b.sw.x
b.sw.y
```

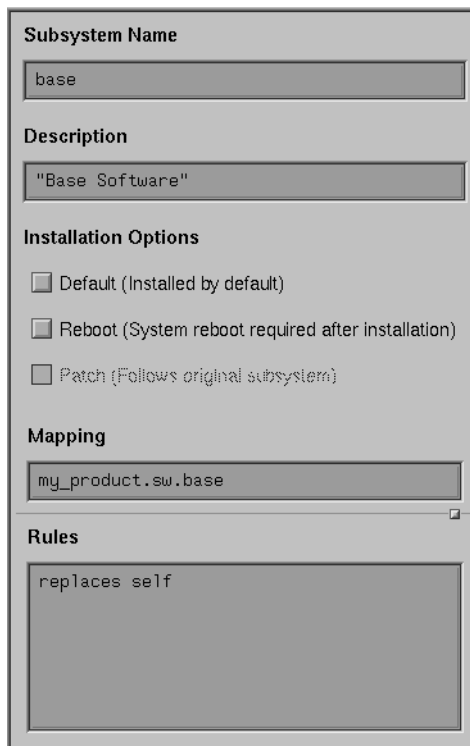
You can control the order in which images are installed. In the example above, for instance, when you package the *b* product you can specify that the subsystems in the *b.sw* image should be installed before the subsystems in the *a.sw* image.

In the example above, to force *b.sw* to be installed before *a.sw*, give the *b.sw* image an order number less than 9999. The installation order of the subsystems would then be:

```
b.sw.x
b.sw.y
a.man.x
a.man.y
a.sw.x
a.sw.y
b.man.x
b.man.y
```

## Entering Subsystem Specifications

The Subsystem Specification sheet, shown in Figure 3-8, is visible whenever a subsystem node is selected in the Product Hierarchy Graph. For example, if you click the left mouse button on the subsystem labeled *optional* in the Product Hierarchy Graph, the Subsystem Specification sheet for the *optional* subsystem appears to the right of the graph.



The screenshot shows a form titled "Subsystem Specification Sheet" with the following sections:

- Subsystem Name:** A text field containing "base".
- Description:** A text field containing "Base Software".
- Installation Options:** Three checkboxes, all of which are unchecked:
  - Default (Installed by default)
  - Reboot (System reboot required after installation)
  - Patch (Follows original subsystem)
- Mapping:** A text field containing "my\_product.sw.base".
- Rules:** A text area containing "replaces self".

**Figure 3-8** The Subsystem Specification Sheet

Use the Subsystem Specification sheet to enter the subsystem's name, description, installation options, and mapping and rules information (each of these text fields is discussed in detail in this section).

To select installation options, just click the left mouse button on the check button next to the option you want to select, then click the *Assign* arrow

button. **If you don't click the *Assign* arrow button, your changes will be lost when you select a different node.**

You can select as many installation options as you wish for each subsystem, but be careful not to select any of these options unnecessarily. For more information, see "Setting Basic Installation Options for the Subsystem."

### **Naming the Subsystem**

To change the name of the selected subsystem, follow these steps:

1. Delete the existing text in the Subsystem Name text field.
2. Type in the new text.
3. Click the *Assign* arrow button.

Refer to "Naming Images and Subsystems" on page 39 for guidelines.

### **Changing the Description of the Subsystem**

To change the description of the selected subsystem, follow these steps:

1. Delete the existing text in the Description text field.
2. Type in the new description.
3. Click the *Assign* arrow button.

**Note:** Always enclose your description within quotes. You can use either single or double quotes. You can use one type of quotes as part of your description if you use the other type of quotes to quote the description. For example, 'Acme 1/2" Tape Support' is a valid description as long as you enclose it in single quotes.

The subsystem description is a brief description of the image. Since this description appears in Software Manager listings, it's important that the description be informative and distinct from the product and image descriptions.

Like the product and image descriptions, the subsystem description *must* begin with the marketing name of the product. This is because Software Manager lists the products, images, and subsystem according to these

descriptions—if your product, image, and subsystem names begin differently, they aren't listed together. (The image description also appears in *versions* listings.)

Here are some general guidelines for creating descriptions:

- Limit the string to 30 characters, if possible. Anything larger gets wrapped or truncated by Software Manager and *versions*.
- When abbreviating words, use only standard, easily understood abbreviations.
- Capitalize the first letter of each word, except prepositions.
- Do not use punctuation.
- Do not include part numbers or marketing codes.

### **Setting Basic Installation Options for the Subsystem**

The Installation Options check buttons that appear in the Subsystem Specification sheet represent three installation options. The first two options—*Default* and *reboot*—are user selected. The third option—*Patch*—is selected automatically when you create a patch through the “Create Patch...” menu item in the File menu. You can select none, one, or all of these options.

#### **The *Default* Option**

The *Default* option specifies that each time a product is installed, some or all of the subsystems are automatically selected for installation (these subsystems are called “default” subsystems). Users can deselect these subsystems if they choose. If you have a lot of subsystems, some of which are not necessary for the average user, you might want to tag the essential subsystems as default.

To set this option, click the left mouse button with the cursor over the option. After making your selection, click the *Assign* button to assign the option(s) to the selected subsystem.

#### **The *Reboot* Option**

The *Reboot* option specifies that the machine must be rebooted after installation of the subsystem. This option is only for subsystems that involve

kernel code or other software that cannot be installed normally. **Use the *Reboot* button only when absolutely necessary!**

To set this option, click the left mouse button with the cursor over the option. After making your selection, click the *Assign* button to assign the option(s) to the selected subsystem.

**Note:** When using Software Manager to install your product, this option is shown as a flag type of 'b'.

### The *Patch* Option

The *Patch* option specifies that a subsystem is part of a patch product and must have a follows rule. *swpkg* automatically assigns the *Patch* option to all subsystems that are part of a patch product. This option is insensitive to user selection. See Chapter 8, "Creating a Patch Product."

### Setting Installation Rules

The Rules text field allows you to specify, for each subsystem in your product, those subsystems that:

- are prerequisites for installation
- are incompatible with the selected subsystem
- should be replaced by the selected subsystem

In order to make these specifications, you need to use the special syntax required for writing these rules. The syntax is described in the sections below and examples are provided.

To use the Rules text field, follow these steps:

1. Select the appropriate subsystem node in the Product Hierarchy Graph.
2. Type the correct rules specifications into the Rules text field. (The syntax is described in the sections below.)
3. Click the *Assign* arrow button.



## Adding Replacement Statements

You can tell Software Manager to remove a specified subsystem when it installs the currently selected subsystem. Essentially, this means telling Software Manager to replace one subsystem with another, usually newer, subsystem. You do this using the *replaces* statement.

Replaces statements are powerful and flexible—taking the time now to learn how they work will help you later when you need to make changes to your product. The replaces line is critical when a product is repackaged (for example, when files in the product get moved around, when subsystems get new names, and so on).

A replaces line simply specifies that you want to replace an older subsystem with a newer subsystem. Specify the name of the subsystem and the range of version numbers you want replaced. Here's the format of the replaces statement:

```
replaces name lowvers highvers
```

or

```
replaces self
```

where

*name* is the name of the subsystem that is going to be replaced.

*lowvers* is the lower boundary of the range of versions of *name* that should be replaced. It can be 0, or any version number value that you supply.

*highvers* is the higher boundary of the range of versions to be replaced. *highvers* can be one of the following:

- *oldvers*, defined as the current version minus 1
- an actual version number that you supply

*self* Software Manager always assumes that a subsystem replaces an older subsystem of the same name. Since this is the default case, you do not need to specify it explicitly.

You can specify as many replaces statements as you need.

For each subsystem that has a replaces line, Software Manager looks to see if the subsystem specified in the replaces line is installed and if its version falls in the range given in the replaces line. If the installed version is in the range, the new subsystem is selected for installation automatically. If the new subsystem is installed, the old version is removed automatically.

There are four typical ways to use replaces:

1. Specify that a subsystem replaces older versions of itself.
2. Specify that a subsystem replaces maintenance versions of itself.
3. Specify that a subsystem replaces different subsystems that are now obsolete.
4. Specify replacement directions in complex repackaging situations where one subsystem has become several or several older subsystems have been restructured into several new ones.

Here are some examples of replaces statements:

A subsystem gets a name change. Suppose that the subsystem *rfind.man.rfind* is split into two subsystems, *rfind.man.client* and *rfind.man.server*. To make sure that *rfind.man.rfind* is replaced properly, you could create a replaces statement that looks like this:

```
replaces rfind.man.rfind 0 oldvers
```

Alternately, you could specify an exact old version number so that your replaces statement would look something like this:

```
replaces rfind.man.rfind 0 1006000106
```

Or, suppose two or more subsystems get repackaged into a single subsystem. For example, say the *rfind.sw.client* subsystem and the *rfind.sw.server* subsystem are combined and the new subsystem is called *rfind.sw.rfind*. To replace the old client and server, you would use these replaces statements:

```
replaces rfind.sw.client 0 oldvers
```

```
replaces rfind.sw.server 0 oldvers
```

If either of the old subsystems *rfind.sw.client* or *rfind.sw.server* is installed, *rfind.sw.rfind* will replace them.

Finally, suppose you want to replace a maintenance release of *rfind.sw.client* with a new base release. You need to replace both the maintenance release and the previous base release. Your replacement statements would look like this:

```
replaces maint.rfind_sw.client 0 oldvers
replaces rfind.sw.client 0 oldvers
```

### Setting Incompatibilities

You can specify that a subsystem in your product can be installed on a user's workstation only if one or more other subsystems are not installed. This is known as specifying incompatibilities.

**Note:** Do not specify incompatibilities unless absolutely necessary.

When incompatibilities are specified, Software Manager does not allow users to install subsystems that are incompatible. It checks for incompatibilities at two different times:

- When a subsystem is selected for installation, Software Manager determines whether or not it is incompatible with something that has already been installed.
- When the user quits, Software Manager checks again among the subsystems it has just installed for incompatibilities.

The format for incompatibility statements is the same as for replaces statements. Specify the name of the subsystem and the range of version numbers you want to declare incompatible. The format of the incompatibilities statement is:

```
incompat name lowvers highvers
```

where

*name* is the name of the subsystem that is incompatible.

*lowvers* is the lower boundary of the range of versions of *name* that are incompatible. It can be 0, or any version number value that you supply.

*highvers* is the higher boundary of the range of versions that are incompatible. *highvers* can be one of the following:

- *maxint*, the maximum value that a long int can hold
- *oldvers*, defined as the current version minus 1
- an actual version number that you supply

Here are some examples of incompatibility statements:

To specify that the *rfind.man.rfind* subsystem is incompatible with the selected subsystem, you could create a incompatibility statement that looks like this:

```
incompat rfind.man.rfind 0 oldvers
```

Alternately, you could specify an exact old version number so that your incompatibility statement would look something like this:

```
incompat rfind.man.rfind 0 1006000106
```

### Setting Prerequisites

You can specify that a subsystem in your product can be installed on a user's workstation only if one or more other subsystems are also installed. That is, you can set prerequisites for the installation of a subsystem.

**Do not specify subsystems outside your product as prerequisites unless absolutely necessary.** Prerequisites can be a real problem, because the prerequisite product can change names or versions, making your prerequisite statement obsolete—which can prevent installation of your product!

To set prerequisites for a subsystem, first select the appropriate subsystem node in the Product Hierarchy Graph. Then type the text specifying the prerequisites into the Rules text field.

You can specify a list of subsystems that must all be installed in order for users to install the new subsystem:

```
prereq (  
    name lowvers highvers  
    name lowvers highvers
```

```

    name lowers highvers
  )

```

where

*name* is the name of the subsystem that is a prerequisite.

*lowvers* is the lower boundary of the range of versions of *name*. It can be 0, or any version number value that you supply.

*highvers* is the higher boundary of the range of versions of *name*. *highvers* can be one of the following:

- maxint, the maximum value that a long int can hold
- oldvers, defined as the current version minus 1
- an actual version number

You can also write a prereq that specifies that only *one* of the listed subsystems must be installed for the new subsystem:

```

prereq (
    name lowers highvers
)
prereq (
    name lowers highvers
)

```

Remember to press the *Assign* arrow button to update the subsystem node after filling in the text area.

Here are some examples of prerequisite statements:

To specify that the *rfind.man.rfind* subsystem is a prerequisite for the selected subsystem, you could create a prerequisite statement that looks like this:

```
prereq rfind.man.rfind 0 oldvers
```

Alternately, you could specify an exact old version number so that your prerequisite statement would look something like this:

```
prereq rfind.man.rfind 0 1006000106
```

To specify that the *rfind.man.server* subsystem *and* the *rfind.man.client* subsystem must *both* be installed in order for the selected subsystem to be installed, you could write a prerequisite statement that looks like this:

```
prereq (  
    rfind.sw.server 0 oldvers  
    rfind.sw.client 0 oldvers  
)
```

To specify that *either* the *rfind.man.server* subsystem *or* the *rfind.man.client* subsystem must be installed in order for the selected subsystem to be installed, you would need to write *two* prerequisite statements. They might look like this:

```
prereq (  
    rfind.sw.server 0 oldvers  
)  
prereq (  
    rfind.sw.client 0 oldvers  
)
```

### Editing Mapping Expressions Directly

This field allows you to write your own mapping expressions. It is highly unlikely that you'll need to use this field, since you can create almost any product structure and mapping using the *swpkg* GUI. However, in cases where you have very complicated existing IDB files that you need to modify, you might find it easier to create mapping expressions rather than restructure your product.

If you decide you want to write your own mapping expressions, refer to Appendix A, "Writing Mapping Expressions."

**Note:** If you use any mapping expression other than a single subsystem name, enclose the expression within quotes. You can use either single or double quotes.

### The Assign Arrow Button

Clicking the *Assign* arrow button saves the changes you made in the *currently visible* Product, Image, or Subsystem Specification sheet. If you

switch to a different Specification sheet before clicking the *Assign* button, your changes are lost.

For example, if you select a product node, then change the product name, then click the *Assign* button, the new product name is applied to the product node. On the other hand, if you select a product node, then change the product name, then select an image node without first clicking the *Assign* button, your new product name is *not* applied to the product node.





## **Tagging the Files**

Once you have created your product hierarchy, you need to tag the files you want to install and assign them to the different subsystems.



---

## Tagging the Files

After you create a product hierarchy using the Create Product Hierarchy worksheet, you need to assign each of your product's files to a subsystem in your product hierarchy. This is called tagging the files. You tag the files using the Tag Files worksheet.

This chapter contains these sections:

- “Tagging the Files: Before You Begin” on page 69 provides some background information and lists the prerequisites for tagging the files.
- “Tagging the Files: The Basic Steps” on page 71 lists the basic steps for tagging your files.
- “Using the Tag Files Worksheet” on page 72 describes the features of the Tag Files worksheet and explains how to use the worksheet to tag your files.

### Tagging the Files: Before You Begin

This section lists the prerequisites for tagging your files, and defines tags and IDB files.

#### Prerequisites

Before you begin tagging the files, you must first create a product hierarchy using the Create Product Hierarchy worksheet (see Chapter 3).

#### What's an IDB File?

*swpkg* stores the information you enter in the Tag Files worksheet in an installation database file (IDB file).

The IDB file contains the following basic information for each file in your product:

- its location in a built source tree
- its location after installation
- its owner, group, and mode
- what subsystem (group of files) it should be packaged in

An IDB file can include directories, links, and FIFOs as well as files.

The IDB file may also list certain attributes to be associated with each file, including:

- the instruction that a binary file not be stripped before including it in the software distribution (the default is to strip binaries automatically)
- shell commands that are to be executed just before or after installation of the file (preops and postops)
- shell commands that are to be executed when exiting Software Manager (exitops)
- a specification that this file is to be installed only on workstations that have certain architectural characteristics, such as a particular type of processor or graphics
- an indication that a previous version of this file may have been modified by users, and if so, how to install this file

### What's a Tag?

*swpkg* requires that you tag each of your files with the name of the subsystem to which you want that file to belong. *swpkg* stores all of these tags in the IDB file and uses them to build the product.

**Note:** You can also create arbitrary tags, and map them to specific subsystems using the mapping expressions described in Appendix A, "Writing Mapping Expressions." This is almost never a good idea, but it can be useful when you need to modify extremely complicated IDB files.

## Tagging the Files: The Basic Steps

You can add the following items to your IDB file:

- files
- directories (when you add a directory, all of its contents are added)
- linked files
- linked directories

Using the Tag Files worksheet, as shown in Figure 4-1, follow these steps to enter files and directories into your product's IDB file and to assign them a subsystem:

1. Add all the files and directories in your product to your IDB file. (Remember to add the *.ftr* and *.fti* files for your Desktop icon.)
  - Use the File Browser to select your file(s) and directory(ies).
  - Click the *Add* arrow button to add them to your IDB file.

The items appear in the IDB file list.

2. Select items from the IDB file list for tagging. You can select multiple items, as long as you're going to put them all in the same subsystem.
3. From the Tags Browser, select the subsystem in which you want the selected items to appear. Then click the left mouse button on the *Assign* arrow button.
4. Repeat Steps 2-3 until every item in your product is assigned to a subsystem.
5. Save the IDB file by opening the File menu and selecting "Save IDB."

**Caution:** Do not include two files with the same full pathname in two different subsystems. If a file is included in two subsystems and both subsystems are installed, and one of the subsystems is subsequently removed, then the common file disappears, leaving a "hole" in the other subsystem. This rule applies only to files, not directories.

## Using the Tag Files Worksheet

This section describes the features of the Tag Files worksheet, shown in Figure 4-1, and explains how to use the worksheet to tag your files.

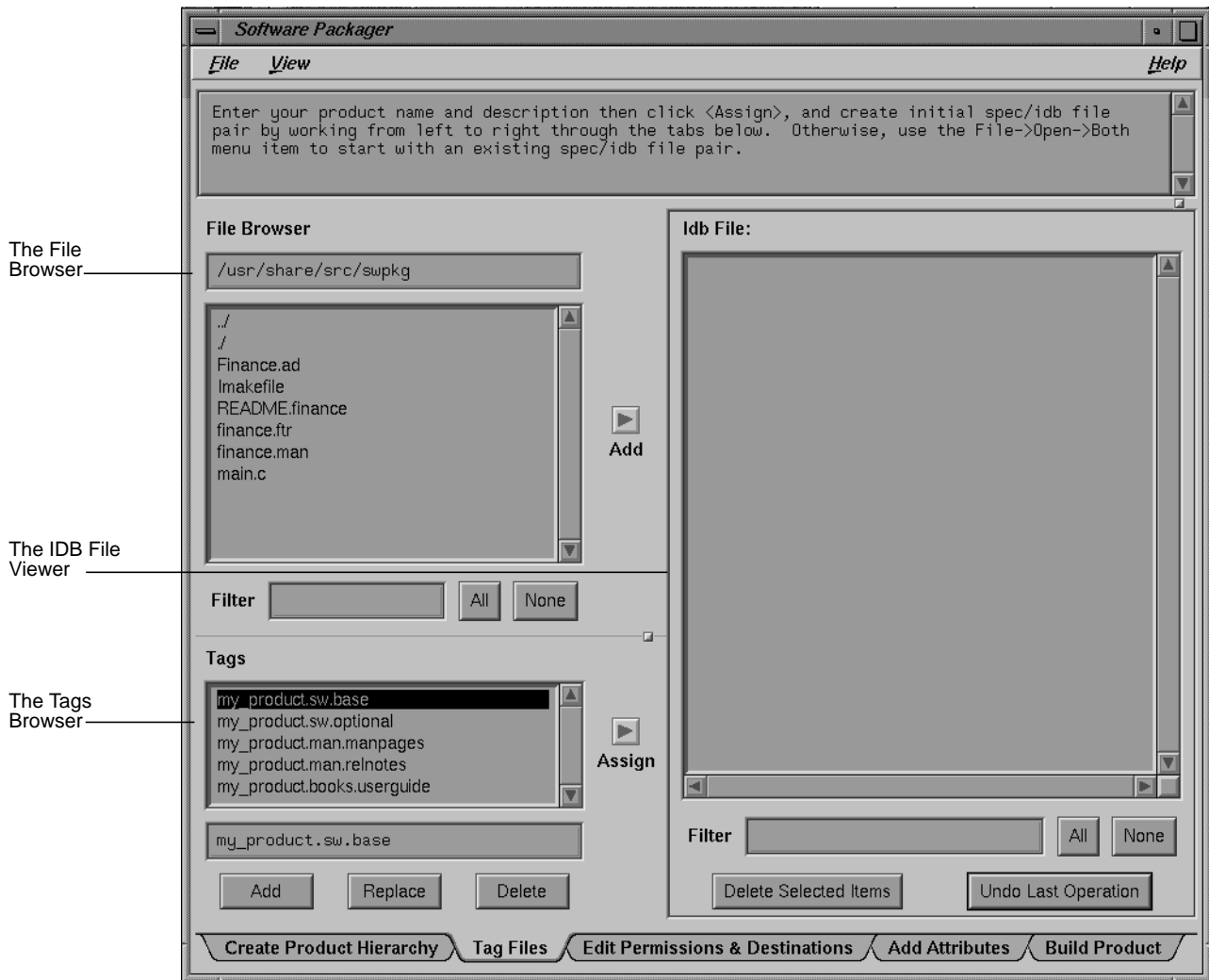
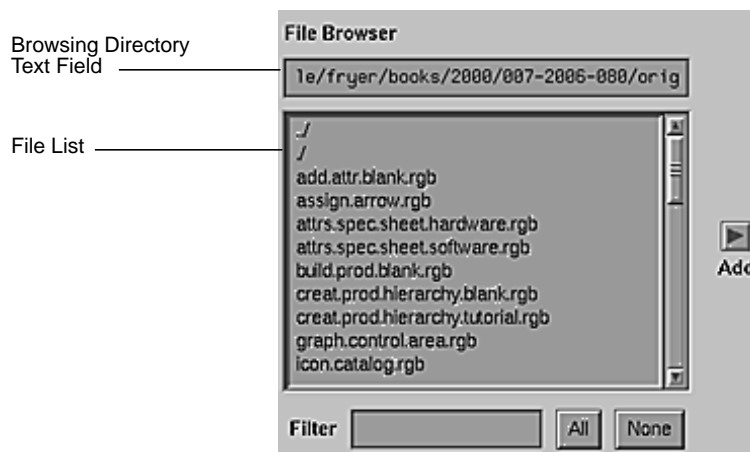


Figure 4-1 The Tag Files Worksheet

## Selecting Product Files Using the File Browser

The File Browser, shown in Figure 4-2, is a tool that lets you select directories and files (including linked directories and files) to add to the IDB file. (You must add all the files in your product to the IDB file.) Use the File Browser to scroll through all the files in a directory and select the ones you need by clicking the left mouse button on the filename. Once you've made your selections, you add them to the IDB file by clicking the *Add* arrow button. The directories and files appear in the IDB File Viewer. When you add a directory, you add the directory and everything in it including subdirectories.



**Figure 4-2** The File Browser

The File Browser consists of the browsing directory text field (which specifies the browsing directory), the file list (a list of all the files and directories in the browsing directory), the Filter text field, and the *All* and *None* buttons.

### Setting the Browsing Directory

The File Browser lists the contents of the *browsing directory*, the directory that is named in the text field at the top of the File Browser. The initial browsing directory is the current directory, provided it's within the source root directory (the default is /).

You can select a different directory either by typing it in the text field and pressing **<Enter>** or, if you want to select a directory that is within the current directory, by double-clicking the left mouse button on the directory of your choice.

You can move up to the parent directory, by double-clicking the line with two periods followed by a slash (`../`).

You can enter your `$HOME` directory by typing:

`~`

in the File Browser text field and pressing **<Enter>**.

### Selecting Files and Directories From the File List

To select a file or directory (which includes its contents) from the File Browser's file list, click the left mouse button on it. You can select multiple contiguous entries by pressing and dragging the mouse before releasing the left button. You can select multiple noncontiguous entries by holding down the **<Ctrl>** key when pressing the left mouse button.

### Selecting Files Using Filters

You can also select items in the file list by typing a pattern in the Filter text field, then pressing **<Enter>**. You can specify any regular expression (see the *regcmp(3G)* reference page for details).

For example, if you type:

`.*`

you select everything in the file list (this is equivalent to clicking the *All* button).

If you type:

`^$`

you deselect everything in the file list (this is equivalent to clicking the *None* button).



### **Selecting Files With the *All* and *None* Buttons**

Click the *All* button to select all the files in the file list. Click the *None* button to deselect all the files in the file list.

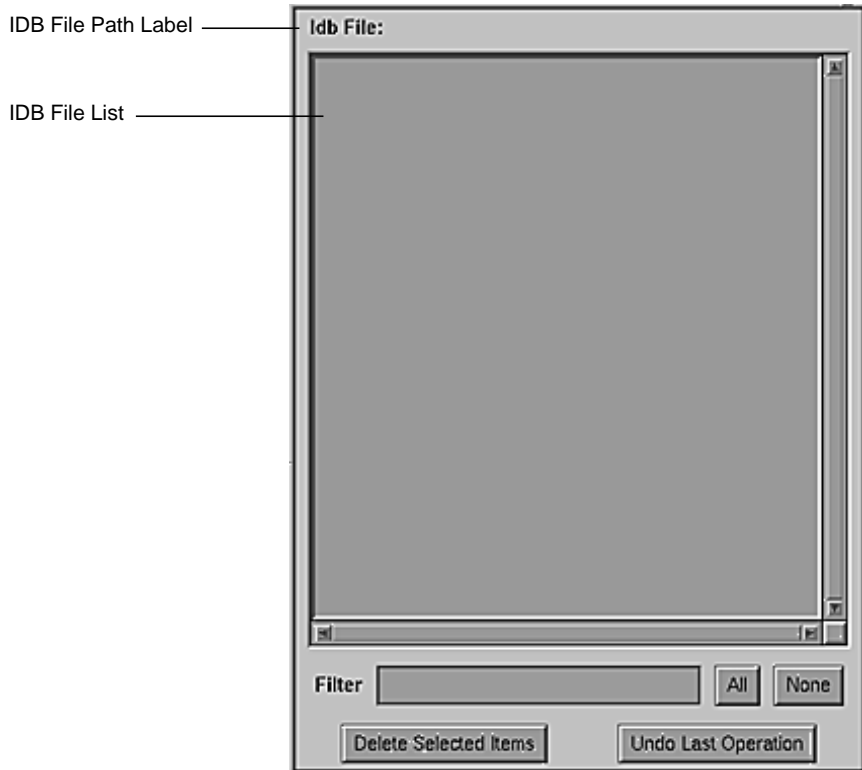
### **Adding Selected Files**

Use the *Add* arrow button to add files and directories to the IDB file list. To do this, select the files and directories from the File Browser's file list, then click the left mouse button on the *Add* arrow button.

### **Accessing Your IDB File Using the IDB File Viewer**

The IDB File Viewer, shown in Figure 4-3, lets you easily scroll through a list of all the files and directories in your product's IDB file and allows you to select and edit each item.

Symbolic links are followed in the Add operation; adding a link to a directory adds the link, the directory and its contents. To avoid adding the directory and its contents, rename the target directory for the duration of the operation.



**Figure 4-3** The IDB File Viewer

The IDB File Viewer consists of the IDB file path label, the IDB file list, the Filter text field, the *All* button, the *None* button, the *Delete Selected Items* button, and the *Undo Last Operation* button.

When you first open the Tag Files worksheet, the IDB file list will probably be empty. You must add each file and directory in your product to the IDB file list by selecting the them from the File Browser, then clicking the *Add* arrow button.

Once all the files and directories in your product are included in the IDB file list, you need to:

- tag each file and directory, using the Tag Files worksheet

- set permissions and destinations for each file and directory, using the Edit Permissions & Destinations worksheet
- set installation attributes for each file and directory, if necessary, using the Add Attributes worksheet

### **IDB File Path Label**

The IDB file path label shows the current IDB file pathname. Until a valid IDB file is identified, no path is listed. For information on creating an IDB file or changing the IDB file path, see “Using the File Menu” on page 5. For a definition of an IDB file, read “What’s an IDB File?” on page 69.

### **Interpreting the IDB File List**

The IDB file list displays the contents of the current IDB file. It lists all the files and directories that appear in your product’s IDB file and, for each file and directory, it displays the associated tags, permissions, destinations, and attributes.

### **Selecting Items From the IDB File List**

To select an item from the IDB file list, click the left mouse button on it. You can select multiple contiguous items by pressing and dragging the mouse before releasing the left button. You can select multiple noncontiguous items by holding down the <Ctrl> key when pressing the left mouse button.

### **Making an Item Current in the IDB File List**

You can make an item in the IDB file list current, by double-clicking it. When you make an item current, *swpkg* fills in the rest of the worksheet with the current settings for that item. This allows you to examine the information conveniently and to change some of the information for an item without having to fill in all the fields manually.

### **Selecting Items Using Filters**

You can select items in the IDB file list by typing a pattern in the Filter text field, then pressing <Enter>. You can specify any regular expression (see the *regcmp*(3G) reference page for details).

### Selecting Items Using the *All* and *None* Buttons

Click the *All* button to select all the items in the IDB file list. Click the *None* button to deselect all the items in the IDB file list.

### Deleting Items From the IDB File List

You can delete an item from the IDB file list by selecting the item, then clicking the *Delete Selected Items* button. You can delete more than one item at a time.

### Undoing Operations

You can undo your last operation by clicking the *Undo Last Operation* button. This restores the worksheet to the state it was in before you performed your last operation. If you change your mind, you can redo the operation by clicking the *Undo Last Operation* button again.

### Selecting Tags Using the Tags Browser

The Tags Browser, shown in Figure 4-4, allows you to easily select a tag from the list of available tags and assign it to any of the items listed in the IDB File Viewer. Each tag corresponds to a subsystem in your product.

Basically, you first select a group of items from the list of items in the IDB File Viewer, use the Tags Browser to select the subsystem to which those items should belong, then click the *Assign* arrow button to assign the selected tag to the selected items.

The Tags Browser consists of the *tags list* (which lists all the available tags), the *tag text field* (which allows you to enter the name of a new tag), the *Add* button, the *Replace* button, and the *Delete* button.



**Figure 4-4** The Tags Browser

### Selecting Tags From the Tags List

The tags list displays a list of all the tags defined in the spec file. These tags correspond to the subsystems you created using the Create Product Hierarchy worksheet. The selected tag appears in inverse video (the letters are in the background color and surrounded by a black box). To select a tag, click it with the left mouse button.

To select all the items in the IDB File List that are tagged with a particular tag, double-click the left mouse button on that tag.

### Editing the Tags List

You can edit the tags list using the tags text field in conjunction with the *Add*, *Replace*, and *Delete* buttons.

It is almost never necessary or desirable to edit the tags list. Instead, change the list of subsystems using the Create Product Hierarchy worksheet. In general, you edit the tags list only when you need to set up a complex mapping of files to subsystems. **If you do edit the tags list, any new tag you create must be mapped to one of the subsystems in your spec file.** Refer to Appendix A for instructions on creating such a mapping.

Here are the ways in which you can edit the tags list:

- To add a new tag, type the new tag into the tags text field, and click the *Add* button (or just press **<Enter>**).

- To replace an existing tag with a new one, select the tag you want to replace, type the new tag into the tags text field, and click the *Replace* button.
- To remove a tag from the list, select it, then click the *Delete* button.

## **Editing Permissions and Destinations**

After tagging your files, you need to set the destination, names, and permissions of the files installed.





---

## Editing Permissions and Destinations

This chapter explains how to set the correct permissions for each file in your product; how to specify where Software Manager should install each file in your product; and how to specify the source tree root.

This chapter contains these sections:

- “Editing Permissions and Destinations: Before You Begin” on page 83 provides some background information and lists the prerequisites for editing permissions and destinations.
- “Editing Permissions and Destinations: The Basic Steps” on page 84 lists the basic steps for editing permissions and destinations.
- “Using the Edit Permissions & Destinations Worksheet” on page 85 explains how to use the worksheet to edit the permissions and destinations for the files in your product.

### **Editing Permissions and Destinations: Before You Begin**

This section lists the prerequisites for editing permissions and destinations, and briefly explains what is meant by “permissions and destinations” in the context of this worksheet.

#### **Prerequisites**

Before you begin editing permissions and destinations for your files, you must create a product hierarchy using the Create Product Hierarchy worksheet (see Chapter 3) and tag the files using the Tag Files worksheet (see Chapter 4).

In particular, make sure that all your product’s files are listed in the IDB Viewer. If some are missing, go back to the Tag Files worksheet and add

them to the IDB file (see “Tagging the Files: The Basic Steps” on page 71 for instructions). You can’t edit permissions and destinations unless your files are listed in an IDB file because that is where *swpkg* stores them. (Every file that is currently listed in your IDB file is listed in the IDB File Viewer.)

### What Are Permissions and Destinations?

Permissions, in the context of this worksheet, means the permissions of the files in your product as they are installed on your users’ workstations—how you want Software Manager to set the file permissions when your users install your product’s files.

Destinations refers to the pathnames of the files in your product as they are installed on your users’ workstations—where you want Software Manager to put the files when your users install them.

### Editing Permissions and Destinations: The Basic Steps

By default, permissions and destinations are picked up from your source files—if your source files have the same pathnames and permissions that you want them to have when installed on the users workstations, then you don’t need to edit the permissions and destinations fields. If there are differences, or if you want to set a source tree root, then you need to edit this worksheet.

To fill out the Edit Permissions & Destinations worksheet:

1. Set the source tree root for all your files, if you want it to be something other than the default (/).
2. Click the *Apply* arrow button.
3. Select a file from the IDB file viewer by single-clicking it.  
**Tip:** If you double-click a file, *swpkg* selects the file and also displays current settings for permissions, source, and destinations.
4. Set the mode for the file.
5. Set the owner for the file.
6. Set the group for the file.

7. Click the *Assign* arrow button.
8. Set the destination directory, if necessary. Click the *Assign* arrow button.
9. Set the destination filename, if necessary. Click the *Assign* arrow button. (See “Setting the Destination Filename” for instructions.)
10. Repeat steps 3-8 for each file listed in the IDB File Viewer.
11. Save the IDB file by selecting Save IDB from *swpkg*'s File menu.

**Caution:** If you don't click the *Assign* button after editing the text fields, your changes will be lost when you select a new file from the IDB file list.

## Using the Edit Permissions & Destinations Worksheet

This section provides an overview of the Edit Permissions & Destinations worksheet, shown in Figure 5-1, and explains how to use the worksheet's features.

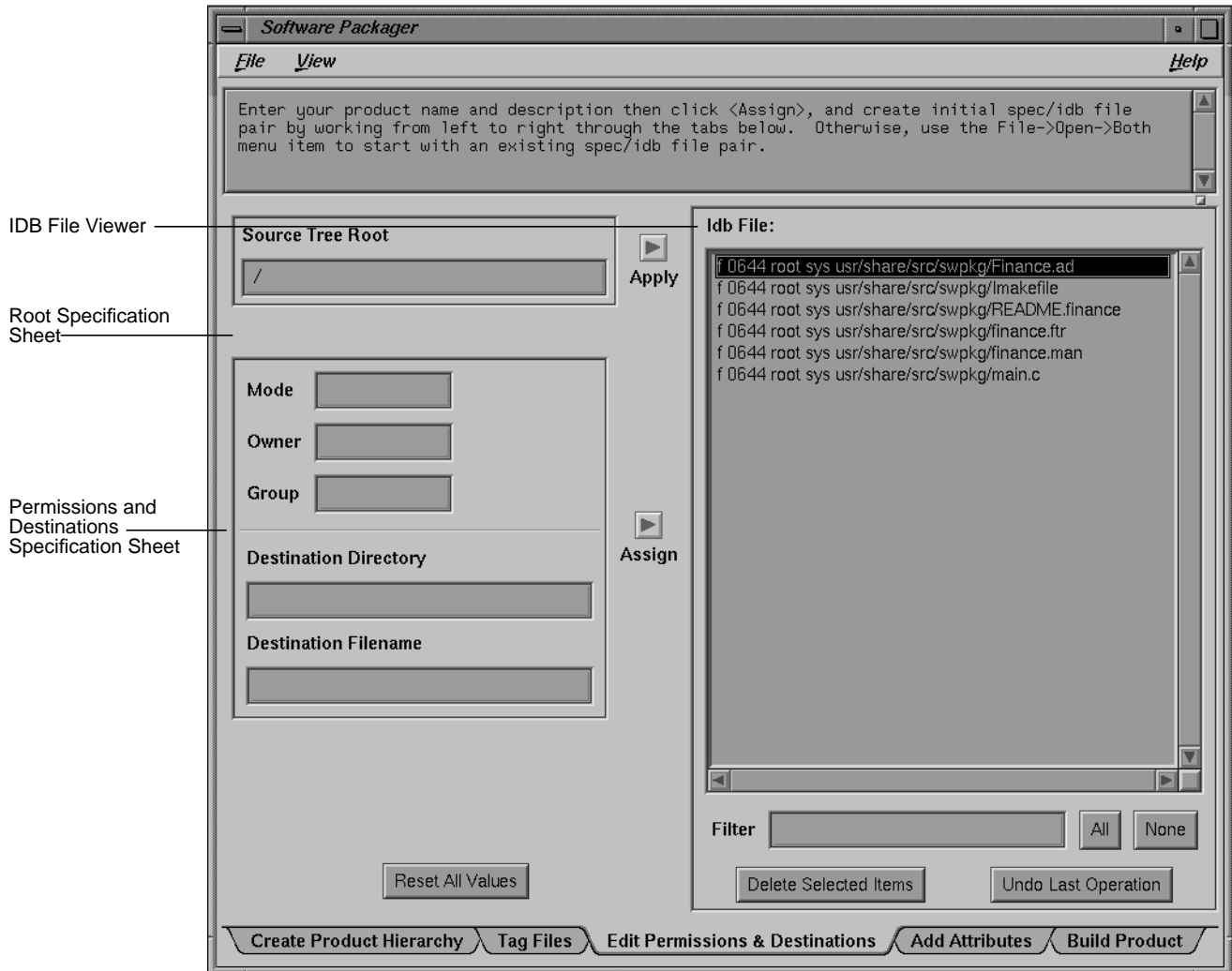
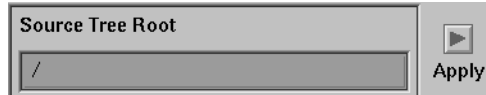


Figure 5-1 The Edit Permissions & Destinations Worksheet

### Setting Source Tree Roots

The Root Specification sheet, shown in Figure 5-2, allows you to specify a root for your source trees.



**Figure 5-2** The Root Specification Sheet

The source tree contains all of the files that you want to include in your packaged product. The source tree root is the directory relative to which *swpkg* looks for your product's files on your workstation (the workstation on which you're building your product). Changing the source tree root limits the browsing area in the Tag Files view to the designated tree. *swpkg* strips the source tree root from the pathnames of all files that you include in your product.

**Caution:** *swpkg* does not save source tree roots for you from session to session. It uses the specified root when you build the product (by clicking the *Build All* button in the Build Product worksheet), but when you quit *swpkg*, the tree root is not saved. Next time you run *swpkg*, you have to set the tree root again. If you forget, then *swpkg* assumes the tree root should be set to / and it will not be able to find your files.

### Setting a Source Tree Root

Roots are always applied to all entries in the IDB file. To set a source tree root for all the files in your product, type the desired source tree root path in the Source Tree Root text field and click the *Apply* arrow button.

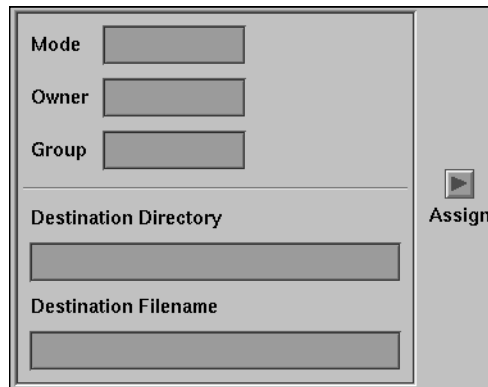
If the source tree root you specify is invalid, then you will see one of the following dialogs when you build the product:

- No files found under Source Root /newroot.
- 10 (of 20 entries) not found under Source Root /root.

**Note:** Instead of the numbers 10 and 20, you see numbers appropriate for your product. The number of files found varies—*swpkg* might think it has found some files but they are probably the wrong ones.

### Setting Permissions and Destination Directories

Use the Permissions and Destinations Specification sheet, shown in Figure 5-3, to set the mode, owner, group, destination directory, and destination filename for each file in your product.



**Figure 5-3** The Permissions and Destinations Sheet

**Note:** If your source files have the same pathnames and permissions that you want them to have when installed on the users workstations, then you don't need to edit the permissions and destinations fields.

#### Specifying the Mode

Use the Mode text field to set the mode of a file as it will be installed on your users' workstations. To set the mode for a file, first make the file current by double-clicking it in the IDB file list. The current settings for the file appear in the text fields. Edit the text in the Mode text field. When you've finished editing, click the *Assign* arrow button.

The mode of a file is an octal number that represents the permissions of the file. An ASCII representation of the mode is displayed to the right of the text area for reference.

Briefly, an absolute mode is given as an octal number constructed from the OR of the following modes:

04000            set user ID on execution

020#0	set group ID on execution if # is 7, 5, 3, or 1, enable mandatory locking if # is 6, 4, 2, or 0. This bit is ignored if the file is a directory; it can be set or cleared only by using the symbolic mode.
01000	sticky bit
0400	read by owner
0200	write by owner
0100	execute (search in directory) by owner
0070	read, write, execute (search) by group
0007	read, write, execute (search) by others

See *chmod(1)* and *ls(1)* for more details on modes.

### Specifying the Owner

Use the Owner text field to specify the owner of a file as it will be installed on your users' workstations. To set the owner for a file, first make the file current by double-clicking it in the IDB file list. The current settings for the file appear in the text fields. Edit the text in the Owner text field. When you've finished editing, click the *Assign* arrow button.

You can use the login ID (for example: root, sysadm, diag, adm, daemon, uucp, nuucp, bin, and guest) or the numerical user ID (uid) to identify who Software Manager should make owner of the file.

For example, entering nothing or 0 is the same as typing in:

```
root
```

In the Owner text field, the uid automatically appears to the right of the login ID for reference.

### Specifying the Group

Use the Group text field to specify the group of a file as it will be installed on your users' workstations.

To set the group for a file, first make the file current by double-clicking it in the IDB file list. The current settings for the file appear in the text fields. Edit the text in the Group text field. When you've finished editing, click the *Assign* arrow button.

You can use the group ID (for example: `sys`, `root`, `daemon`, `bin`, `adm`, `mail`, `uucp`, `lp`, `user`, and `guest`) or the numerical group ID (`gid`) to identify the group to which Software Manager should assign the file.

For example, entering nothing or 0 is the same as typing in:

`sys`

In the Group text field, the `gid` automatically appears to the right of the group ID for reference.

### **Setting the Destination Directory**

Use the Destination Directory text field to set the directory in which Software Manager should install the selected file(s) on your users' workstations.

To set the destination directory for a file, follow these steps:

1. Make the file current by double-clicking it in the IDB file list. The current settings for the file appear in the text fields.
2. Edit the text in the Destination Directory text field. Click the *Assign* arrow button.

### **Setting the Destination Filename**

Use the Destination Filename text field to set the name of each file in your product as you want Software Manager to install it on your users' workstations.

To set the destination filename for a file, follow these steps:

1. Make the file current by double-clicking it in the IDB file list. The current settings for the file appear in the text fields.
2. Edit the text in the Destination Filename text field.
3. Click the *Assign* arrow button.



### **Resetting All Text Fields to the Default Values**

Click the *Reset All Values* button, to reset all the text fields in the worksheet to their default values.



## **Adding Attributes**

Software Packager allows you to set attributes for the files installed, including commands to execute before or after installation, modifying configuration files, and installing specific files on systems with only certain hardware configurations.



---

## Adding Attributes

This chapter explains how to set certain installation options, called attributes, for the files in your product. It contains these sections:

- “Adding Attributes: Before You Begin” on page 95 provides some background information and lists the prerequisites for adding installation attributes.
- “Adding Installation Attributes: The Basic Steps” on page 96 lists the basic steps for adding attributes.
- “Using the Add Attributes Worksheet” on page 97 describes the features of the worksheet and explains how to use it to add installation attributes to your product’s files.

### Adding Attributes: Before You Begin

This section lists the prerequisites for adding installation attributes, and explains what installation attributes are.

#### Prerequisites

Before you begin specifying attributes for your products files, you must first have created a product hierarchy using the Create Product Hierarchy worksheet (see Chapter 3), tagged the files using the Tag Files worksheet (see Chapter 4), and edited permissions and destinations for the files using the Edit Permissions & Destinations worksheet (see Chapter 5).

In particular, make sure that all your product’s files are listed in the IDB Viewer. If some are missing, go back to the Tag Files worksheet to tag them and include them in the IDB file (see “Tagging the Files: The Basic Steps” on page 71 for instructions). You can’t add attributes to your product’s files unless they are listed in the IDB file, because that is where *swpkg* stores them.

(Every file that is currently listed in your IDB file is listed in the IDB File Viewer.)

### What Are Attributes?

For each file in your product, you can specify certain *attributes*—installation options that govern the installation of that file. For example, you can specify

- that the installation occur during the next boot
- shell commands for Software Manager to run automatically at different points in the installation
- that a separate copy of your application be installed for diskless client
- that a binary not be stripped
- on which hardware configurations your application can be installed

*swpkg* stores attributes information in your product's IDB file.

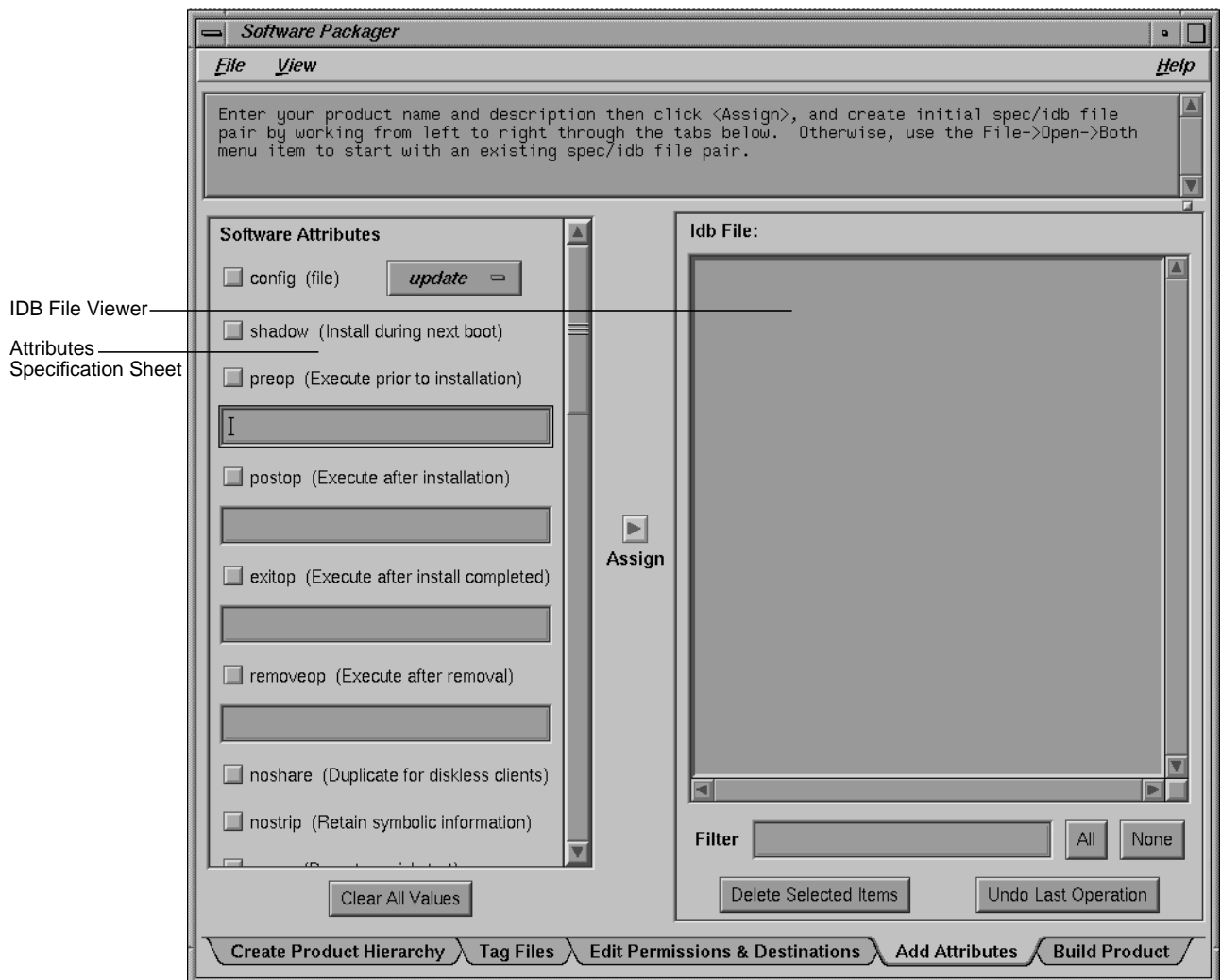
### Adding Installation Attributes: The Basic Steps

To specify attributes for your product's files, follow these steps:

1. Select a file or files from the IDB file list.
2. Select the desired attributes from the Attributes Specification sheet.
3. Click the *Assign* arrow button.
4. Repeat Steps 1-3 for each file or group of files for which you wish to specify attributes.
5. Save the IDB file by selecting Save IDB from *swpkg*'s File menu. (This step is optional; you can wait and save the IDB file when you build the product.)

## Using the Add Attributes Worksheet

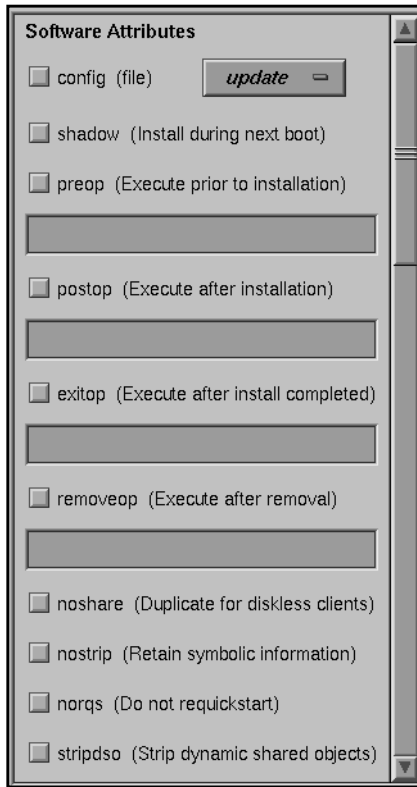
This section describes the features of the Add Attributes worksheet, shown in Figure 6-1, and explains how to use the worksheet to add installation attributes to your product's files.



**Figure 6-1** The Add Attributes Worksheet

### Selecting Software Installation Attributes

Figure 6-2, lists the for Software Installation Attributes. To the left of each attribute is a check button. You select (or deselect) the attribute by clicking the check button. When the attribute is selected, a red check mark appears over the button.



**Figure 6-2** The Attributes Specification Sheet: Software Attributes

#### The config Attribute

Users sometimes modify the configuration files for your product. Typically, these modifications are made to reflect site- and machine-specific information. It's important that you identify all the configuration files in



your product so that modifications users made to older versions of these files are not lost during installation on the new versions.

By checking the config attribute check button, you identify the selected file as a configuration file. You must then specify a configuration type to specify how to deal with each configuration file. There are three configuration types. The first configuration type, update, is listed on the large button located to the right of the config attribute. Click this button to see all three choices, then select the appropriate configuration type from the list.

After you've selected a configuration type, click the *Assign* arrow button to assign the attribute to the selected file.

Here are the three configuration types:

update	Software Manager automatically installs the new configuration file, but saves the old file as <i>filename.O</i> , where <i>filename</i> is the name of the configuration file.
noupdate	Software Manager retains the existing (old) file, if it exists.
suggest	Software Manager retains the existing file, if it exists, and install the new configuration file as <i>filename.N</i> , where <i>filename</i> is the name of the configuration file.

The configuration types are listed below in a table that describes what Software Manager does with each type of configuration file.

**Table 6-1** Configuration Types

Configuration File Type	Before installation	After installation
update	No previous version was installed.	The new version is installed.
	A previous version was installed, but not modified.	The new version is installed.
	A previous version was installed and modified.	The older version is <i>filename.O</i> and the new version is installed.
noupdate	No previous version was installed.	The new version is installed.

**Table 6-1 (continued)** Configuration Types

Configuration File Type	Before installation	After installation
suggest	A previous version was installed, but not modified.	The previous version is still installed.
	A previous version was installed and modified.	The previous version is still installed.
	No previous version was installed.	The new version is installed.
	A previous version was installed, but not modified.	The new version is installed.
	A previous version was installed and modified.	The new version is installed as <i>filename.N</i> .

**The shadow Attribute**

Very few applications need this attribute—in general, don’t select it. Selecting the shadow attribute delays the installation of the specified file until the next time the workstation is booted. So, Software Manager does not immediately install the file on the users’ workstations.

This attribute is useful for products that might be used by a variety of applications at any given time—such as shared libraries—where a replacement of the existing product files might affect those applications that are using the product at the time a new version of the product is installed.

After selecting this attribute, click the *Assign* arrow button to assign the attribute to the selected file.

**The preop Attribute**

Use the *preop* attribute when you want *swpkg* to execute a list of commands before installing a selected file.

See the following sections:

- Steps for Using an Ops Command

- Example: Send Email Before Installing Executable
- Ops Limitations

### **Example: Send Email Before Installing Executable**

The following example sends email to the developer (you) just before the executable is installed by the user:

```
"echo $USER@`hostname`.`domainname` installed my_product| Mail developer@abc.com"
```

You can cut, paste, and edit this example.

### **The postop Attribute**

Use the *postop* attribute when you want *swpkg* to execute a list of commands after installing a selected file. Configuration files cannot have postops.

See the following sections:

- Steps for Using an Ops Command
- Example: Send Email After Installing Executable
- Ops Limitations

### **Example: Send Email After Installing Executable**

The following example sends email to the developer (you) after the executable is installed by the user:

```
"echo $USER@`hostname`.`domainname` installed my_product| Mail developer@abc.com"
```

You can cut, paste, and edit this example.

### **The exitop Attribute**

Use the *exitop* attribute when you want to execute a list of commands after the user quits Software Manager (assuming the user is installing the selected file).

See the following sections:

- Steps for Using an Ops Command

- Example: Installing an Application Icon
- Ops Limitations

### Example: Installing an Application Icon

The following example installs your application's icon into the Icon Catalog:

```
"if [ -x \${$rbase}/usr/sbin/iconbookedit ]; then
chroot \${$rbase} /usr/sbin/iconbookedit -add \"Category:File
Name:/usr/bin/X11/my_product \" -syspage Application; fi"
```

You can cut, paste, and edit this example.

For a more detailed discussion of the *iconbookedit* command, see the *iconbookedit(1M)* reference page or Chapter 11, “Creating Desktop Icons: An Overview” in the *Indigo Magic Desktop Integration Guide*.

**Caution:** Do not use *exitop* to add directories. Doing so can cause the size of your product to change, and because *inst* doesn't know about directories created through *exitop*, it cannot account for the size change. Add directories through the Tag Files worksheet.

### The *removeop* Attribute

Use the *removeop* attribute to specify a list of command for *swpkg* to execute after a file is removed. *removeops* are executed only when a subsystem is removed (not during an upgrade).

See the following sections:

- Steps for Using an Ops Command
- Example: Send Email After Removing Executable
- Ops Limitations

### Example: Send Email After Removing Executable

The following example sends email to the developer (you) after the user removes the executable (e.g. `versions remove my_product`).

```
"echo $USER@`hostname`.`domainname` removed my_product | Mail developer@abc.com"
```

You can cut, paste, and edit this example.

### Steps for Using an Ops Command

To use an ops command, follow these steps:

1. Select a file from the IDB file list.
2. Select the ops attribute.
3. Type commands into the ops text field.

–or–

Cut and paste the example code found in the specific ops section, and replace code as necessary.

If you type the commands, be sure to

- enclose the list of commands in a pair of double quotes
  - precede semicolons with a backslash, if you want them taken literally
  - press <Enter> after each separate command
4. Click the *Assign* arrow button.

**Note:** When saving your ops commands into the IDB file, *swpkg* converts new lines into semicolons. When you start a new session with this IDB file, *swpkg* converts the semicolons back to new lines. To tell *swpkg* that you really do want a semicolon and not a new line, put a backslash in front of all “real” semicolons.

### Ops Limitations

Although an ops attribute provides the flexibility to perform many types of processing at installation time, **you should limit its use for these reasons:**

- If the commands in the list do any type of processing that increases the use of disk space, such as uncompressing files, the user could run out of disk space during installation because Software Manager has no way of knowing how much disk space is required.

- If files in the software product are modified during installation, the *versions -m* command reports them as being modified. This can be confusing for users because they have no way of finding out what the changes were and in many cases won't know that they were done during installation.
- Extensive processing in an ops command makes installations take longer.
- Errors that occur while running an ops command are very hard for users to diagnose and correct.

#### **The noshare Attribute**

When your software product is “installed” on a diskless workstation, each file is installed on a server in the share tree by default. This means that just one copy of each file is installed on the server and that copy is used by all diskless clients of that server. Your product may contain files that need to be duplicated for each client, typically because they are configuration files that must be modified for each client workstation.

To indicate that a file is to be replicated for each diskless client, select the noshare attribute from the Attribute Viewer. After selecting this attribute, click the *Assign* arrow button to assign the attribute to the selected file.

#### **The nostrip Attribute**

By default, *swpkg* automatically strips all binaries. You can override this behavior by selecting the nostrip attribute from the Attribute Viewer. After selecting this attribute, click the *Assign* arrow button to assign the attribute to the selected file. (See the *strip(1)* reference page for more information.)

#### **The norqs Attribute**

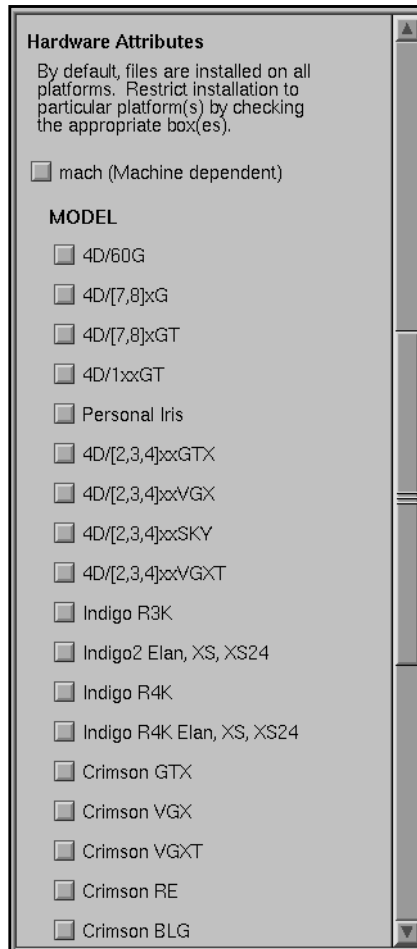
Use the norqs attribute when you don't want an executable to be included on the list for *rqsall* (see the manpage for *rqsall*). For example, add this attribute to any executable you don't want modified as a result of a quickstart.

### **The stripdso Attribute**

Use the stripdso attribute to strip symbolic information from shared library objects.

### **Selecting Hardware Installation Attributes**

The lower portion of the Add Attributes worksheet (see Figure 6-3) titled Hardware Attributes, provides a list of the model names for selection. Clicking on a name selects/deselects it for assignment. Click the *Clear All Values* button to deselect all items.



**Figure 6-3** The Attributes Specification Sheet: Hardware Attributes

In most cases, the files you include in your software product are applicable to all models of Silicon Graphics workstations. However, *swpkg* does provide a mechanism for restricting the installation of files to particular models on a per-file basis. You can use this mechanism to prevent users from installing files on a workstation that can't run them (which is better than allowing users to install files that won't work properly).



You'll need to set up installation restrictions if:

- your application doesn't run on all Silicon Graphics workstations
- different versions of a file are required for different models
- a file in your product doesn't apply to particular models

You can restrict installation by specifying model or board names.

### **Restricting Installation to Specific Models**

The lower section of the Add Attributes view labeled Hardware Attributes presents a list of model names for selection. Clicking on a name selects or deselects it for assignment. Clicking on the *Clear All Values* button deselects all items.

To see a list of boards used with specific models, choose "Show SGI Board Names" from the View menu.

To restrict installation to specific models:

1. Select the file(s) from the IDB File List.
2. Select the models on which the selected file(s) can be installed.
3. Click the *Assign* arrow button.

### **Restricting Installation to Specific Boards**

You can restrict installation to specific boards in three categories; CPUBOARD (Central Processing Unit Board), GFXBOARD (Graphics Board), and SUBGR (Subgraphics Board).

Model and board names are interdependent. *swpkg* references a matrix (*/var/inst/machfile*) of legitimate model and board combinations. For example, pressing the *R3k Boards Only* button selects all of the R3000 based cpuboard, and makes R4000 based models insensitive. You can still make further CPU board selections and deselections by clicking the check button corresponding to a particular CPU board.

**Note:** The R4000 boards include derivatives of the R4000 architecture, such as R4400.

To get the values for CPUBOARD, GFXBOARD, and SUBGR for a particular workstation, use the *hinv*(1M) command. From the output, to find out whether the workstation has an R3k or an R4k processor, look at the line that starts with "CPU." For more information on the type of CPU board, look at the first line of the *hinv* output.

The *hinv* output provides graphics board and subgraphics board information using a different terminology than is used in the GFXBOARD and SUBGR sublists. To determine the values for GFXBOARD and SUBGR from *hinv* output, first look at the line labeled "Graphics board." The *IRIS Software installation Guide* provides a table that maps the *hinv* output for each type of graphics board to the corresponding GFXBOARD and SUBGR values.

Since new values of CPUBOARD, GFXBOARD, and SUBGR may have been added for new models of workstations released after the *IRIS Software Installation Guide* was published, always check the most recent *Release Notes for System Software* for a current list of possible values.

### **Assigning the Selected Attributes**

Use the *Assign* arrow button to assign selected attributes to a file or files in the IDB file list. To do this, select the file(s) from the IDB file list, make your selections from the Attributes Specification sheet, then click the left mouse button on the *Assign* arrow button.

## **Building the Product**

This chapter describes how to build the installable product. It also discusses how to change a product that you've built, how to merge two or more existing products into a single product, and how to divide an existing product into two or more separate products.



---

## Building the Product

This chapter explains how to build your product. It contains these sections:

- “Building the Product: Before You Begin” on page 111 provides some background information and lists the prerequisites for building your product.
- “Building the Product: The Basic Steps” on page 112 lists the basic steps for building your product.
- “Using the Build Product Worksheet” on page 113 describes the features of the worksheet and explains how to use it to build your product.
- “After the First Build” on page 118 describes how to change a product that you’ve built, how to merge two or more existing products into a single product, and how to divide an existing product into two or more separate products.

### Building the Product: Before You Begin

This section lists the prerequisites for building a product and explains how *swpkg* builds a product.

#### Prerequisites

Before you build your product, you must first create a product hierarchy using the Create Product Hierarchy worksheet (see Chapter 3), tag the files using the Tag Files worksheet (see Chapter 4), and edit permissions and destinations for the files using the Edit Permissions & Destinations worksheet (see Chapter 5). Also, you must specify installation attributes, if any, for each file in your product (see Chapter 6).

In particular, make sure that all your product's files are listed in the IDB Viewer. If some are missing, go back to the Tag Files worksheet to tag them and include them in the IDB file (see "Tagging the Files: The Basic Steps" on page 71 for instructions).

**Note:** You only need to complete the above steps once for each product (unless you need to change the product hierarchy). After that, you can use the Build Product worksheet to build the product as often as necessary.

### How Does *swpkg* Build a Product?

*swpkg* builds a product using the *gendist* command, which generates the primary components for software products: the product descriptor, the product IDB, and the images.

To build a product, *gendist* needs three things:

- a tree containing all the files to be shipped
- a master IDB file containing a description of each file or directory to be included in the product
- a distribution specification (spec) file that describes the product structure

*swpkg* takes your selections and input for each worksheet and uses them to create the required spec and IDB files for *gendist*. You can generate and/or edit these files by hand, if you like.

For more information on the *gendist* command, see the *gendist(1M)* reference page.

## Building the Product: The Basic Steps

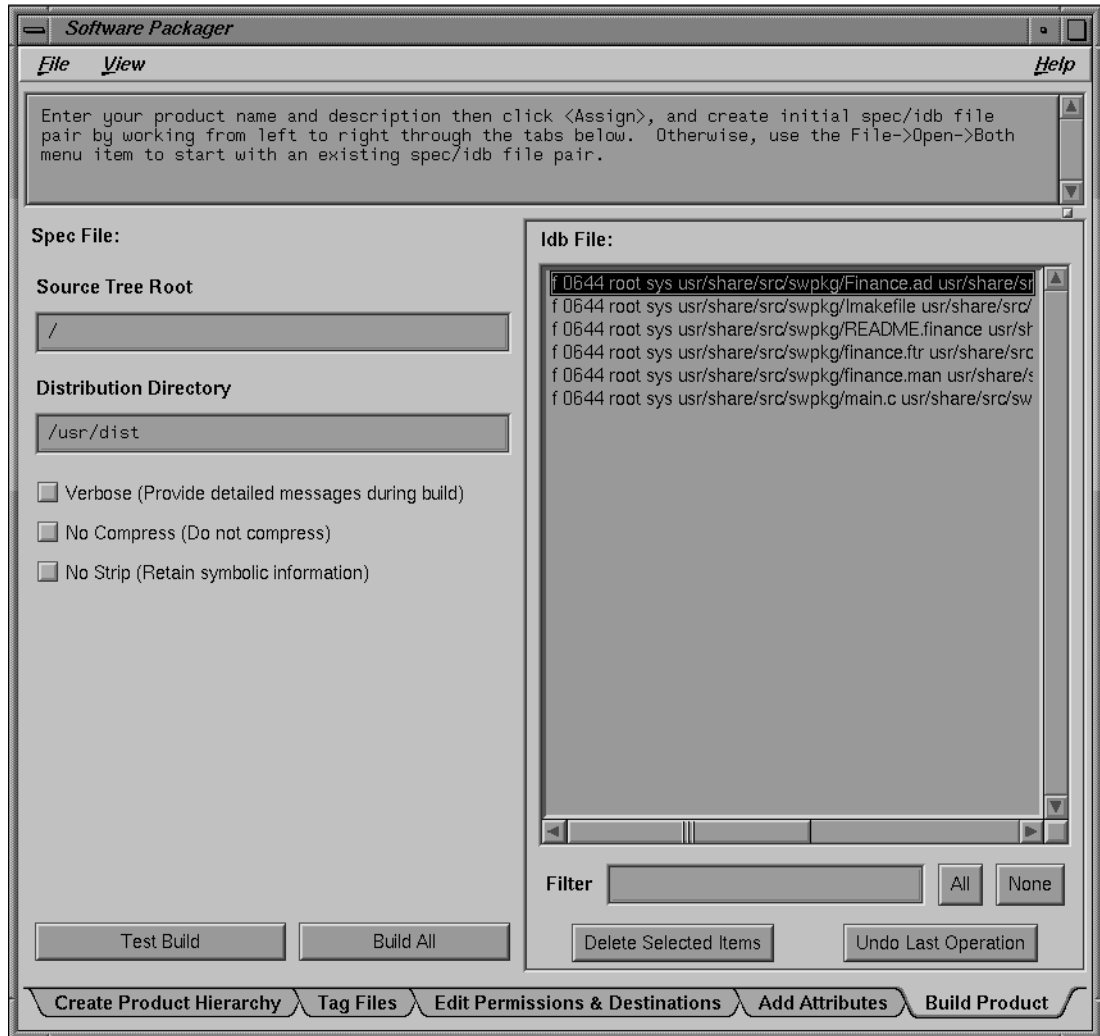
To build a product, follow these steps:

1. Select one or more items from the IDB file list.
2. Specify the desired build options for the selected item(s) by clicking the appropriate Build Options check buttons. (See "Selecting Build Options" on page 116 for descriptions of the build options.)

3. Save the spec and IDB files using the “Save” menu item from the File pull-down menu.
4. Try a test of the build by clicking the *Test Build* button.
5. Resolve any error messages that appear in the Message Area during the test. Appendix B, “Troubleshooting,” provides some troubleshooting information.
6. Build the product by clicking the *Build All* button.

## Using the Build Product Worksheet

This section describes the features of the Build Product worksheet, shown in Figure 7-1, and explains how to use the worksheet to build a product.



**Figure 7-1** The Build Product Worksheet

The Build Product worksheet contains these major parts:

- Worksheet Selection Tabs (described in “Using the Worksheet Selection Tabs” on page 8)
- Message Area (described in “Using the Message Area” on page 8)



- The IDB File Viewer (described in “Accessing Your IDB File Using the IDB File Viewer” on page 75)
- The Spec File Path Label (described in “The Spec File Path Label” on page 115)
- The Source, Destination, and Distribution text fields (described in “Setting Tree Root and the Distribution Directory” on page 115)
- The Build Options check buttons (described in “Selecting Build Options” on page 116)
- The *Test Build* button (described in “Running a Test Build” on page 118)
- The *Build All* button (described in “Building the Product” on page 118)

### **The Spec File Path Label**

The spec file path label shows the current spec file pathname. Until a valid spec file is identified, no path is listed. For information on creating a spec file or changing the spec file path, see “Using the File Menu” on page 5. For a definition of a spec file, read “What’s a Spec File?” on page 40.

### **Setting Tree Root and the Distribution Directory**

If you haven’t already specified a source root in the Edit Permissions & Destinations worksheet, you can do so using the Source Tree Root text field. It is not necessary to specify this tree root. You do, however, need to specify the distribution directory for your product. This section explains how to set the tree root and the distribution directory.

#### **Setting a Source Tree Root**

Typically, you’ll have already specified the source tree root in the Edit Permissions & Destinations worksheet. You can also set the source tree root here. Before changing the source tree root, you should read “Setting Source Tree Roots” on page 86, which tells you what the source tree root is and warns you of the risks involved.

To set a source tree root, type the source tree root path in the Source Tree Root text field and press <Enter>.

If the source tree root you specify is invalid, then you will see one of the following dialogs when you build the product:

- No files found under Source Root /newroot.
- 10 (of 20 entries) not found under Source Root /root.

**Note:** Instead of the numbers 10 and 20, you see numbers appropriate for your product. The number of files found varies—*swpkg* might think it has found some files but they are probably the wrong ones.

### Setting the Distribution Directory

The distribution directory is the directory in which *swpkg* puts the built, installable product files. The default distribution directory is */usr/dist*.

To set a distribution directory for all the files in your product, first select all the files in your product by clicking the *All* button in the IDB File Viewer). Then type your desired distribution directory path in the Distribution Directory text field and press **<Enter>**.

### Selecting Build Options

The Build Options check buttons allow you to select any or all of these four build options for each of your product's files:

- Verbose
- Maint
- No Compress
- No Strip

To select an option, just click the left mouse button on the appropriate option check button. The option is selected when a red check mark appears on the button.

**The *Verbose* Build Option**

Check the *Verbose* button when you want *swpkg* to work in verbose mode—providing more output as the distribution is created. This output appears in the Message Area.

**The *Maint* Build Option**

Checking the *Maint* button tells *swpkg* to generate a maintenance product. Silicon Graphics recommends that you do not use this option at this time. It is included in *swpkg* for compatibility with the previous tools, but it might be replaced in future releases. In general, rather than creating a maintenance product, it's better to create a new version of your product.

A maintenance product contains only files that include bug fixes, new features, or support for new hardware. When users install a maintenance release, the files in the maintenance release overwrite existing versions of those files. If a previously installed file does not have a replacement file in the maintenance release, it is not removed.

A maintenance release can include files from many products, but is packaged as one or sometimes two products. Product names for maintenance releases are usually of the form "maint" followed by a digit (this digit has no inherent meaning). Image names are created by taking the original product names and image names and joining them with an underscore (\_) rather than a period. Subsystem names remain the same. For example, the maintenance version of the subsystem *oe1.sw.unix* is named *maint1.oe1\_sw.unix*.

**The *No Compress* Build Option**

Check the *No compress* button to tell *swpkg* not to compress the images being built.

**The *No Strip* Build Option**

Check the *No Strip* button to tell *swpkg* not to strip any of the executables.

## Running a Test Build

The *Test Build* button initiates a dry run, which reads the files and checks their validity without writing anything. Error messages, if any, appear in the Message Area. Appendix B, “Troubleshooting” provides some troubleshooting information.

## Building the Product

The *Build All* button builds the product—creating the files in the Distribution Directory. These include binary versions of the IDB and spec files. Error messages, if any, appear in the Message Area.

## After the First Build

After you’ve built your product the first time, subsequent builds for that product are much easier, since you already have a spec and IDB file. This section discusses subsequent *swpkg* builds. It contains these sections:

- “Building the Product After the First *swpkg* Build” explains how to build a product that’s been built (using *swpkg*) before.
- “Combining Existing Products Into a Single Product” explains how to take two existing products that have both been built using *swpkg* and merge them into a single product.
- “Incorporating the Help Subsystem into a Product” on page 123 describes how to merge an online help subsystem with your product.
- “Breaking an Existing Product Into Two Products” explains how to break apart an existing product (that’s been built with *swpkg*) into two or more products.

**Note:** If you have existing spec and IDB files that were not generated with *swpkg*, you can use them with *swpkg*. However, you’ll have to remove any *includes* or *defines* from the IDB file, because *swpkg* doesn’t handle them.

**Warning:** Do not copy files from the source directory into the distribution directory: you will be overwriting important files. Your spec and idb files are saved in the source directory. When your product is built, *gendist* saves files of the same name in the distribution directory. Though these files are identically named, they are not the same.

## Building the Product After the First *swpkg* Build

This section explains what to do under these circumstances:

- You've already built your product at least once using *swpkg* and you don't want to make any changes to the spec and IDB files (for example, changing filenames, adding new files, or changing the product hierarchy).
- You've used *swpkg* to build your product once, but you want to make some changes to the spec and IDB files.

The first case is the easiest. After making changes to the files that comprise your product (fixing bugs, adding features, and so on), you want to build the product again. Since you haven't changed any of the filenames and you don't want to alter the product hierarchy, you can go straight to the Build Product Worksheet. Follow these steps:

1. Start *swpkg* and open the Build Product worksheet.
2. From the File menu, open the "Open" cascade menu and select "Both." The Open Spec window appears.
3. Use the file browser and text field to select the spec file for your product and click the *OK* button. The Open Idb window appears.
4. Use the file browser and text field to select the IDB file for your product and click the *OK* button.
5. If, in the previous build of your product, you specified a source tree root, enter the root in the corresponding text field now. Source roots are discussed in "Setting Source Tree Roots" on page 86. (*swpkg* does not save the source tree root in the spec and IDB files, so you have to re-enter this information each time you build the product.)

6. Select the desired build options. The build options are described in “Selecting Build Options” on page 116. (*swpkg* does not save build option selections in the spec and IDB files, so you have to re-enter this information each time you build the product.)
7. Try a test of the build by clicking the *Test Build* button.
8. Resolve any error messages that appear in the Message Area during the test. Appendix B, “Troubleshooting,” provides some troubleshooting information.
9. Build the product by clicking the *Build All* button.

If you do need to change the spec and/or IDB file, then you’ll have to make those changes before you build the product again. The exact steps depend on the changes you want to make, but in general:

1. Start *swpkg*.
2. From the File menu, open the “Open” cascade menu and select “Both.” The Open Spec window appears.
3. Use the file browser and text field to select the spec file for your product and click the OK button. The Open Idb window appears.
4. Use the file browser and text field to select the IDB file for your product and click the OK button.
5. Use whatever worksheets you need in order to make the desired changes. For example, if you just want to change installation attributes for a particular file or files, then open the Add Attributes worksheet, make your changes, and build the product.

If you also need to add a new file, then you’ll need to open the Tag Files worksheet to tag the file and the Edit Permissions and Destinations worksheet to set the permissions and destinations for the new file.

If you need to set installation attributes for the new file, you’ll need to use the Add Attributes worksheet as well.

6. Open the Build Product worksheet.
7. If, in the previous build of your product, you specified a source tree root, enter that root in the corresponding text field now. Source tree roots are discussed in “Setting Source Tree Roots” on page 86. (*swpkg* does not save the source tree root in the spec and IDB files, so you have to re-enter this information each time you build the product.)

8. Select the desired build options. The build options are described in “Selecting Build Options” on page 116. (*swpkg* does not save build option selections in the spec and IDB files, so you have to re-enter this information each time you build the product.)
9. Try a test of the build by clicking the *Test Build* button.
10. Resolve any error messages that appear in the Message Area during the test. Appendix B, “Troubleshooting,” provides some troubleshooting information.
11. Build the product by clicking the *Build All* button.

### **Combining Existing Products Into a Single Product**

This section explains how to merge two existing products (that were both originally built with *swpkg*) into a single product. You can merge more than two products, if you like, by appending more than one spec and IDB file to the spec and IDB files for the first product (see Step 2). To merge two existing products into a single product, follow these steps:

1. Start *swpkg*.
2. From the File menu, open the “Open” cascade menu and select “Both.” The Open Spec window appears.
3. Use the file browser and text field to select the spec file for the first product and click the *OK* button. The Open Idb window appears.
4. Use the file browser and text field to select the IDB file for the first product and click the *OK* button.
5. From the File menu, open the “Append” cascade menu and select “Spec.” The Open Spec window appears. Use the file browser and text field to select the spec file for the second product and click the *OK* button.
6. From the File menu, open the “Append” cascade menu and select “Idb.” The Open Idb window appears. Use the file browser and text field to select the IDB file for the second product and click the *OK* button.

7. Now you have a single spec file and a single IDB file. Each file contains everything that was in each of the corresponding files for the two original products. Use the Create Product Hierarchy to set up a product hierarchy for the new product, choose a new product name, and so on.
8. Use the Tag Files worksheet to make sure each filename is tagged correctly.
9. If the two original products used different source tree roots, open the Edit Permissions and Destinations worksheet and correct the pathnames so that they agree with whatever source tree root you're going to use for the new product. Set the source tree root for the product, if you don't plan to use the default values (/).
10. If you want to change permissions, destinations, or installation attributes for a particular file, open the appropriate worksheet and make your changes.
11. Open the Build Product worksheet. If the two original products used the same source tree root, enter that root in the corresponding text field now (if you didn't already do this in Step 9). Source tree roots are discussed in "Setting Source Tree Roots" on page 86. (*swpkg* does not save the source tree root in the spec and IDB files, so you have to re-enter this information each time you build the product.)
12. Select the desired build options. The build options are described in "Selecting Build Options" on page 116. (*swpkg* does not save build option selections in the spec and IDB files, so you have to re-enter this information each time you build the product.)
13. Try a test of the build by clicking the *Test Build* button. When you're asked where to save the Spec and IDB files, choose new filenames so that you don't overwrite the spec and IDB files for the original products.
14. Resolve any error messages that appear in the Message Area during the test. Appendix B, "Troubleshooting," provides some troubleshooting information.
15. Build the product by clicking the *Build All* button.



## Incorporating the Help Subsystem into a Product

If you've created online help for your product as described in Chapter 9, "Providing Online Help With SGIHelp," in the *Indigo Magic Desktop Integration Guide*, you should incorporate the help into your installable images. "Producing the Final Product" in that chapter describes how to create an installable help subsystem, which you should do before incorporating it with the rest of your product. That process automatically creates appropriate spec and IDB files for the help subsystem; tags the files; sets the permissions and destinations; and assigns the necessary attributes. The tools that create the online help subsystem use "/" as the Source Tree Root directory.

If you've already created the spec and IDB files for your product using *swpkg*, you can merge the help subsystem with the existing files as described in "Combining Existing Products Into a Single Product" on page 121.

If you've not already created the spec and IDB files for your product, you can open the existing help subsystem spec and IDB files, and expand them as needed to handle the rest of your product.

## Breaking an Existing Product Into Two Products

To break apart an existing product (built with *swpkg*) into two or more products, follow these steps:

1. Start *swpkg*.
2. From the File menu, open the "Open" cascade menu and select "Both." The Open Spec window appears. Use the file browser and text field to select the spec file for the product and click the OK button. The Open Idb window appears. Use the file browser and text field to select the IDB file for the product and click the OK button.
3. Open the Create Product Hierarchy worksheet and delete nodes from the Product Hierarchy graph until you have the structure you want for the first of your two "new" products.
4. Open the Tag Files worksheet and delete all the files that don't belong in the first product. Re-tag any remaining files, if necessary.

5. Use the other worksheets to make any other changes necessary to the remaining files (such as changing the product name, changing installation attributes, and so on).
6. If, in the previous build of the (whole) product, you specified a source tree root, enter that root in the corresponding text field now. Source tree roots are discussed in “Setting Source Tree Roots” on page 86. (*swpkg* does not save the source tree root in the spec and IDB files, so you have to re-enter this information each time you build the product.)
7. Select the desired build options. The build options are described in “Selecting Build Options” on page 116. (*swpkg* does not save build option selections in the spec and IDB files, so you have to re-enter this information each time you build the product.)
8. Try a test of the build by clicking the *Test Build* button. When you’re asked where to save the Spec and IDB files, choose new filenames so that you don’t overwrite the spec and IDB files for the original product.
9. Resolve any error messages that appear in the Message Area during the test. Appendix B, “Troubleshooting,” provides some troubleshooting information.
10. Build the first of the two new products by clicking the *Build All* button.
11. Go back to the Create Product Hierarchy worksheet and open the Spec and IDB files for the original (whole) product.
12. Repeat Steps 2 through 10 for the second of your two “new” products.

## **Chapter 1**

### **Creating a Patch Product**

This chapter describes how to create a patch product.



---

## Creating a Patch Product

This chapter explains how to create a patch product.

### Creating a Patch Product: The Basic Steps

The following steps show you how to create a patch product. For more information about a patch product, see “Patch Product Requirements and Concepts.”

1. Open the existing product files — both the spec and idb files.
  - From the File menu, select “Open.”
  - From the rollover menu, select “Both.”
2. Create the patch.
  - From the File menu, select “Create Patch...”
  - Provide a product name, or keep the default.
  - In the Tag/Source list, select the files to be included in the patch. Use <Shift>click for continuous selection and <Ctrl>click for random selection.
  - Click OK.

This step creates both the spec and idb files for the patch product.
3. Keep or edit the follow rule.
  - Click on a subsystem in the Create Product Hierarchy worksheet.
  - Keep or edit the follows rule in the Subsystem Specification sheet (see “Patch Product Requirements and Concepts”)
4. Keep or edit the permissions, destinations, and attributes using the appropriate worksheets.

**Note:** When you create a patch through the “Create Patch...” menu item in the File menu, swpkg automatically copies all of the permissions, destinations, and attributes from the original files into the patch product and eliminates the need to reset these items.

5. Build the patch product (see Chapter 7, “Building the Product”).
6. If you’ve forgotten to add a subsystem, begin the process again.

Because all of the settings in the original files are automatically copied when you create a patch through the “Create Patch...” menu item on the File menu, you may find it easiest to start the process over in order to include additional subsystems.

If you choose to add a subsystem through the Create Product Hierarchy worksheet, you must write a follows rule for the subsystem and set the permissions, destinations, and attributes.

## Patch Product Requirements and Concepts

A patch product involves the following concepts and requirements.

- The patch product is separate from the original product.
- The original product files are not modified.
  - Even though you open the original product files, they are not modified when you create the patch. Just before opening the Create Patch window, swpkg closes the original files.
- The default name for the patch product is the original product name followed by `_patch#` where # is a number that increments with each patch created for the product.

**Warning:** You can edit this name. However, SGI recommends that you use the provided name which allows swpkg to increment patches appropriately.

- All subsystems in the patch product are automatically assigned a follows rule. This rule is mandatory (only one is allowed) and can be edited in the Create Product Hierarchy worksheet by clicking on a subsystem node. It has the following format:

`follows name lowers highvers`

where

*name* Is the name of the subsystem being replaced.

*lowvers* Is the lower boundary range of the product versions to be replaced. Use 0 or higher.

*highvers* Is the higher boundary range of the product versions to be replaced. Use one of the following:

oldvers, interpreted as the current version minus 1

an actual version number that you supply

For example:

If your original product name is *finance*, and you've written a patch for the base software subsystem, version 2, the follows rule would read something like:

```
follows finance.sw.base 2 2
```

- When you save the patch product, you are provided with the default names of *product\_patch#.spec* and *product\_patch#.idb*

where

*product* Is the name of the original product

# Is a number that increments with each patch created for the product.

You can edit this name.





## Writing Mapping Expressions

In most cases, you can simply assign files directly to subsystems using the Tag Files worksheet. Occasionally, though, you might have complex cases where it's simplest to create *mapping expressions* for assigning files to subsystems.



---

## Writing Mapping Expressions

This appendix contains these sections:

- “About Mapping Expressions” on page 133
- “Variables and Data Types” on page 134
- “Operators” on page 134
- “Built in Variables” on page 135
- “Built-In Functions” on page 136
- “Statements” on page 137
- “Example” on page 137

### About Mapping Expressions

By writing effective expressions, you can gather files with different names from anywhere and pull them together into one subsystem, or qualify which files with a particular IDB tag go into a subsystem.

Although you can usually make these changes by changing your product structure and tags using the *swpkg* worksheets, you might occasionally prefer instead to add mapping expressions directly to the spec file. Mapping expressions are typically used when you have a very complex existing spec file in which you need to change the file-to-subsystem mappings—but you do not want to completely overhaul the spec file.

You can change the file-to-subsystem mappings by typing valid mapping expressions into the Mappings text field in the Complete Product Hierarchy worksheet. This appendix explains the “expressions language” used to create valid mapping expressions.

**Note:** Using expressions can cause problems because subsystem names can change, breaking the intended behavior. Silicon Graphics recommends that you introduce expressions carefully, and only when absolutely necessary.

## Variables and Data Types

Values are typed as integer or string—non-zero integers and non-null strings are considered “true” in boolean tests. A reference to an IDB attribute is “true” if the IDB attribute is present in the database record being operated on.

References to IDB attribute arguments are made with a form of subscripting after the argument name, where a list of integers (or integers separated by “..” indicating a range) selects specific arguments. The variable *argc* within brackets refers to the last argument. The selected arguments are concatenated with separating spaces and returned as a string value. (Note that the mechanisms of IDB attribute reference just described are likely to change. They are not terribly useful as is.)

Integer and string variables and constants are available, with single and double quotes being entirely equivalent around string constants.

## Operators

The primary values may be combined with most of the usual operators, which behave as in C unless otherwise noted:

+	add
-	subtract
*	multiply
/	divide
=~	pattern match
!=	pattern not match
//	substring
::	concatenation

<code>&amp;</code>	bitwise and
<code> </code>	bitwise or
<code>^</code>	bitwise exclusive or
<code>~</code>	bitwise (unary) not
<code>&amp;&amp;</code>	logical and
<code>  </code>	logical or
<code>!</code>	logical not
<code>!=</code>	not equal comparison (on integers or strings)
<code>==</code>	equal comparison (on integers or strings)
<code>&lt;=</code>	less than or equal comparison (on integers or strings)
<code>&gt;=</code>	greater than or equal comparison (on integers or strings)
<code>&lt;</code>	less than comparison (on integers or strings)
<code>&gt;</code>	greater than comparison (on integers or strings)
<code>?:</code>	conditional
<code>=</code>	assignment
<code>,</code>	expression list

Parenthesis for grouping are also available.

## Built in Variables

<code>type</code>	The file type as a one-character string; the first character of file, directory, block device, character device, (symbolic) link, or (named) pipe (that is, FIFO).
<code>mode</code>	Permission bits. The type of this value is integer, but is converted to a string according to context (though it's a decimal integer, which is probably not what you want).
<code>owner</code>	The name of the owner. The UID is mapped through <i>etc/passwd</i> .

group	The name of the group. The GID is mapped through <i>etc/group</i> .
dstpath	The relative (to root) pathname of the file in the software product destination tree.
srcpath	The relative pathname of the file in the source tree.
nattr	The integer number of IDB attributes associated with the record being operated on.
argc	(Defined only within IDB attribute argument list references.) The number of arguments for the current IDB attribute.
sbase	The pathname of the root of the source tree.
rbase	The pathname of the root of the destination tree.
IDB	The pathname of the primary IDB file. When files are accessed, the mapping between user and group integer IDs and the owner and group string values in the IDB are based on the <i>etc/passwd</i> and <i>etc/group</i> files, respectively. These are first sought under sbase, then under rbase, then under /.

## Built-In Functions

spath(s)	Returns an absolute pathname for the argument; if the given value is relative, it is concatenated with the value of sbase. This is useful in converting a <i>srcpath</i> value into an absolute pathname.
rpath(s)	Returns an absolute destination pathname, concatenated with the value of rbase.
putrec()	Prints the current record in standard format (that is, on one line, packed).
printf(f,a...)	Formatted print (subset of stdio printf). Recognizes field widths with leading zero pad indicator, types %s, %d, %o.
print(a...)	Unformatted print. Prints values as decimal integers or strings, separated by spaces, terminated with newline.
bytes(s)	Returns the size, in bytes, of the given file, or -1 if not found.

blocks(s)	Returns the size, in blocks, of the given file, or -1 if not found.
access(s,m)	Returns the value of the <i>access(2)</i> system call.

## Statements

The following statements are implemented as in C: if [ else ], while, for, break, continue, return, grouping with braces, and expressions.

## Example

The basic spec file for *rfind* doesn't follow the recommendation that each software subsystem have a matching reference page system. Instead, a single IDB tag was used for all reference pages. You could use this expression to put the .1 reference pages into one subsystem and the .1m reference pages into another:

```
exp 'rfind.man.rfind && srcpath =~ "*.1"'  
exp 'rfind.man.rfind && srcpath =~ "*.1m"'
```





## **Appendix B**

### **Troubleshooting**

This chapter provides tips for troubleshooting common Software Packager problems.



---

## Troubleshooting

This chapter provides these sections:

- “Checklist of Do’s and Don’ts” contains a list of requirements for successful packaging of software.
- “Error Messages” lists some common error messages and makes suggestions for finding and fixing the problems.
- “Other Problems” discusses other problems that you might have when trying to build your product.

### Checklist of Do’s and Don’ts

Recommendations for successful packaging of software have been included throughout this guide. Recommendations that should not be violated are repeated in the list below so that you can easily verify that you haven’t violated important recommendations.

- Don’t include hard links in your product.
- Don’t put a file in more than one subsystem.
- Don’t begin any subsystem name with a digit.
- Begin all product, image, and subsystem descriptions with the marketing name of your product.
- Do not select the *Required* subsystem installation option unless the subsystem is truly required for operation of the workstation.
- Parens are illegal.

## Error Messages

Here's a list of possible errors and problems:

- An error in the spec file prevented *gendist* from parsing it.
- A failure might be a sign that the IDB file was not sorted. It can also mean that *swpkg* cannot read the spec file properly.
- Warnings about duplicate files indicate that there are two or more files with identical pathnames in a subsystem and these files don't have "mach" dependent attributes that specify that there are machine-specific versions of this file. Fix these warnings by making the pathnames for each of the files in your subsystem unique.
- Invalid IDB attributes result in messages.
- Error messages might indicate that you have empty products, images, or subsystems. Check to determine whether or not they need to exist. If not, you can just comment them out temporarily in the spec file.

## Other Problems

This section addresses some of the problems you are most likely to encounter and presents approaches to solving the problem.

### Problem

Obsolete subsystems or older versions of subsystems show up in a *versions -a* listing after installation.

### Meaning

If you install all of the subsystems in a product and a *versions -a* listing shows multiple lines for products, images, and subsystems, then the new version of your product doesn't completely remove the old version of your product.

### Corrective Action

Add replaces statements to the spec file that get rid of older versions and obsolete subsystems. See "Setting Installation Rules" on page 58 for instructions.

**Problem**

Files are missing from subsystems.

**Meaning**

Somewhere along the way, some part of the process failed.

**Corrective Action**

For each file that is missing, check for these conditions:

- The file is listed in the IDB file.
- The IDB tag for the file mapped to a valid subsystem.
- The file doesn't have a mach tag for a machine other than the one you installed on.

**Problem**

Syntax error message.

**Possible Meaning**

You started with an existing product that had *includes* or *defines* in the IDB file. *swpkg* doesn't understand these, so you see a syntax error.

**Corrective Action**

Remove any *includes* or *defines* from the IDB file.

