Software Distributor Administration Guide for HP-UX 11i

HP Computers

Manufacturing Part Number: B2355-90754

June 2002, Edition 3

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notice

Trademark Notices

Publication History

The manual's publication date and part number indicate its current edition. The publication date will change when a new edition is released. The manual part number will change when extensive changes are made.

To ensure that you receive the new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

- *Software Distributer Administration Guide*

  For HP-UX 11i: June 2002, Edition 3, **B2355-90754**

  For HP-UX 11i: June 2001, Edition 2, **B2355-90740**

  For HP-UX 11i: December 2000, Edition 1, **B2355-90699**

- *Managing HP-UX Software with SD-UX*

  For HP-UX 11.00: November 1997, Edition 6, **B2355-90154**

- *Managing HP-UX Software with SD-UX*

  For HP-UX 10.30: April 1997, Edition 5, **B2355-90127**

  For HP-UX 10.20: June 1996, Edition 4, **B2355-90107**

New editions of this manual will incorporate all material updated since the previous edition. For the latest version, see the HP-UX 11i Operating System documentation on the web:

`http://docs.hp.com`

and

`http://software.hp.com/SD_AT_HP`

Please direct comments regarding this guide to:

Hewlett-Packard Company
HP-UX Learning Products
3404 East Harmony Road
Fort Collins, Colorado 80528-9599

Or, use this web form to send us feedback:

`http://docs.hp.com/assistance/feedback.html`

Conventions

We use the following typographical conventions.

audit (5)	An HP-UX manpage. *audit* is the name and *5* is the section in the *HP-UX Reference*. On the web and on the Instant Information CD, it may be a hot link to the manpage itself. From the HP-UX command line, enter "`man audit`" or "`man 5 audit`" to view the manpage. See *man* (1).
Book Title	The title of a book. On the web and on the Instant Information CD, it may be a hot link to the book itself.
Emphasis	Text that is emphasized.
Emphasis	Text that is strongly emphasized.
`ComputerOut`	Text displayed by the computer.
`Command`	A command name or qualified command phrase.
`Computer`	`Computer font` indicates literal items displayed by the computer. For example: `file not found`
`Filename`	Text that shows a filename and/or filepath.
`UserInput`	Commands and other text that you type.
`Variable`	The name of a variable that you may replace in a command or function or information in a display that represents several possible values.
`[ ]`	The contents are optional in formats and command descriptions.
`{ }`	The contents are required in formats and command descriptions. If the contents are a list separated by `|`, you must choose one of the items
`...`	The preceding element may be repeated an arbitrary number of times.
`|`	Separates items in a list of choices.

Contents

1 Introduction to Software Distributor

This chapter contains overview information and explains important concepts that will help you use the SD-UX commands most effectively.

Table 1-1 Chapter Topics

Topics:
"About This Guide" on page 20
"SD-UX Overview" on page 22
"SD-UX Concepts" on page 27
"Using the GUI and TUI Commands" on page 35
"Working from the Command Line" on page 55

About This Guide

This guide describes how to use Software Distributor to install, configure, package, and manage software for HP-UX on HP 9000 systems

This guide is written for:

- Stand-alone HP-UX users, primarily concerned with quick and easy access to the right software management tools to get their normal work done. This may include software installation, viewing, or removal and basic depot management.
- HP-UX system administrators, primarily concerned with keeping other users up and running. Additional software management tasks may include more complex depot management, security, patch management, remote operations, and performance management.
- Software packagers, primarily concerned with packaging software into SD-usable format and writing scripts that may accompany the packaged software.

NOTE The most current version of this document, as well as all Hewlett-Packard documentation, is always found at:

`http://www.docs.hp.com/`

What's new in this edition This edition offers additional depot management examples, improved organization, error corrections, and numerous minor enhancements.

Web papers included here This guide also includes information from this previously published paper:

- *Understanding SD-UX ACLs*

Related Documentation and Training

Check the SD-UX web site often for announcements, updates to the *SD-UX FAQ*, and to download the latest version of SD-UX:

`http://software.hp.com/SD_AT_HP`

For additional information on topics covered in this guide:

- SD-UX FAQ:

  `http://software.hp.com/SD_AT_HP/faq.html`

- For details on SD-UX training included in HP-UX system administration classes:

  `http://software.hp.com/SD_AT_HP`

SD-UX Overview

Software Distributor for HP-UX (SD-UX) provides you with a powerful set of tools for centralized HP-UX software management. When connected by a LAN or WAN, each computer running SD-UX can act as a server, allowing its resources to be managed or accessed by other machines, or as a client, managing or using the resources of other machines.

Software Distributor commands are included with the HP-UX operating system and, by default, manage software on a local host only. You can also enable remote operations, which let you install and manage software simultaneously on multiple remote hosts connected to a central controller.

Note that SD-UX running under HP-UX 11.00 and higher versions does not support NFS diskless clusters.

Network Requirements

- Networked systems must support TCP/IP.
- Because Software Distributor is based on distributed, client/server technology, it requires some networking functionality on the host system for proper execution. These networking services are only available in UNIX Run Level 2 (Multi-User mode) and above. Software Distributor cannot run in Single-User mode.

SD-UX Programs and Commands

The following list provides a brief description of each command and references for more detailed information.

Table 1-2 SD Commands

Command & Manpage	Description/Features	More Information
swinstall (1M)	• Installs or updates software • Optional GUI	• "Installation with swinstall" on page 64
swlist (1M)	• Lists installed software or software in depots or on media • Optional GUI	• "Listing Your Software (swlist)" on page 96 • "Listing Registered Depots" on page 158
swcopy (1M)	• Copies software from one depot to another • Optional GUI	• "Copying Software Depots" on page 137
swremove (1M)	• Removes installed software or software in a depot • Optional GUI	• "Removing Installed Software (swremove)" on page 122 • "Removing Software from Depots" on page 162
swpackage (1M)	• Creates packages of software which can then be used as a source for other SD-UX commands	• Chapter 10, "Creating Software Packages," on page 301
swconfig (1M)	• Runs configuration scripts on installed software • Configures, reconfigures, or unconfigures	• "Configuring Your Installation (swconfig)" on page 82 • Chapter 11, "Using Control Scripts," on page 369

Table 1-2 SD Commands (Continued)

Command & Manpage	Description/Features	More Information
swask (1M)	• Runs interactive request scripts that gather information for later use by swinstall or swconfig	• Chapter 11, “Using Control Scripts,” on page 369 • “Requesting User Responses (swask)” on page 407
swacl (1M)	• Specifies, lists, and changes Access Control Lists (ACLs) for SD security.	• Chapter 9, “SD-UX Security,” on page 255
swverify (1M)	• Verifies the integrity of installed software or depot software by comparing IPD information with the files actually installed • Runs verify and fix scripts	• “Verifying Your Installation (swverify)” on page 89 • “Verifying a Depot (swverify -d)” on page 161
swmodify (1M)	• Modifies the Installed Products Database (IPD) and various catalog files that contain information about the software on the system	• “Modifying the IPD (swmodify)” on page 115
swreg (1M)	• Registers newly created depots to make them visible to other systems	• “Registering and Unregistering Depots (swreg)” on page 151
sd (5)	• Starts the Job Browser GUI to create, monitor, schedule, and delete jobs • Requires that remote operations are enabled	• Chapter 6, “Remote Operations Overview,” on page 189 • Chapter 7, “Using Jobs and the Job Browser,” on page 215

Table 1-2 SD Commands (Continued)

Command & Manpage	Description/Features	More Information
swjob (1M)	• Monitors jobs from the command line • Requires that remote operations are enabled	• "Monitoring Jobs from the Command Line" on page 230 • Chapter 6, "Remote Operations Overview," on page 189 • Chapter 7, "Using Jobs and the Job Browser," on page 215
install-sd (1M)	• Re-installs SD-UX from media	• Appendix C, "Replacing or Updating SD-UX," on page 479
swagentd (1M)	• Daemon for SD-UX commands • Must be scheduled before a system is available as a destination for SD-UX commands	• See manpage

The sd, swinstall, swcopy, swlist, and swremove commands each have an optional Graphical User Interface (GUI) with windows and pull-down menus. The GUI commands also work on text-based terminals, providing a Terminal User Interface (TUI), which uses the keyboard instead of the mouse for screen navigation.

You can invoke all SD-UX commands and programs from the command line. The syntax, options, defaults and operands are similar for all commands. See "Working from the Command Line" on page 55 for more information.

SD-UX Online Documentation

To view the a manpage for each command, type:

`man` *command_name*

For additional technical information, type:

`man 5 sd` for SD-UX overview

`man 4 sd` for file layouts

`man 4 swpackage` for packaging file layouts

SD-UX Concepts

Understanding SD-UX concepts, terms, and model of software management will help you use the commands and programs most effectively. For additional definitions, see the Glossary.

Important Terminology

Host refers to any system on which software is to be installed or managed using the SD-UX commands. A **local host** is the system on which you invoke SD-UX commands.

When you have enabled remote operations, you can use SD-UX to operate on one or more **remote hosts**—a host other than the system on which the SD-UX command has been invoked. (See Chapter 6, “Remote Operations Overview,” on page 189 for more information on remote operations.)

A **controller** is the SD-UX program or command (swinstall, swcopy, etc.) that you invoke on your system. The controller may work with data or start processes on other systems.

A **depot** is a repository of software products that can be managed by SD-UX. A depot consists of either a (specially formatted) directory, or physical media such as tapes, CD-ROMs or DVDs. (CD-ROM and DVD depots are really just special instances of directory depots). Directory depots are useful because you can access them via a network. They are often used to store collections of software copied from other depots.

In general, the term **target** refers to either a host (specifically, the host’s file system) or a depot that resides on a host. The term **source** refers to a depot from which software is being installed or copied (sometimes referred to as a **source depot**).

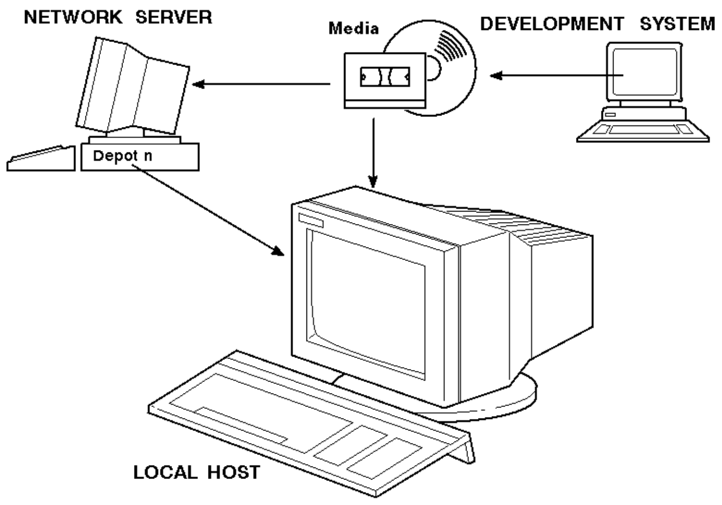
For example, a basic install operation with the swinstall command involves installing software from a source depot to a target location on the host itself. The source depot might be physical media accessible from the target, or a directory depot on some server on the network. The target host might be the same host on which the command was invoked (i.e., the local host) or, if remote operation is enabled, some other host on the network.

A basic copy operation (using the swcopy command) is very similar, except that the target is a depot on the host, rather than the host itself.

For most operations, controller programs access hosts and depots using an agent called swagent, which performs the basic software management tasks. The agent is accessed via a daemon called swagentd. When SD-UX operates on the local host, both controller and agent run on the local host. For remote operations, the agent runs on a remote host.

Figure 1-1, “SD-UX Systems,” shows how software can be developed and then packaged into SD-formatted media, which can either be accessed directly or copied to a depot directory on a server and accessed via the network.

Figure 1-1 **SD-UX Systems**



Software Structure

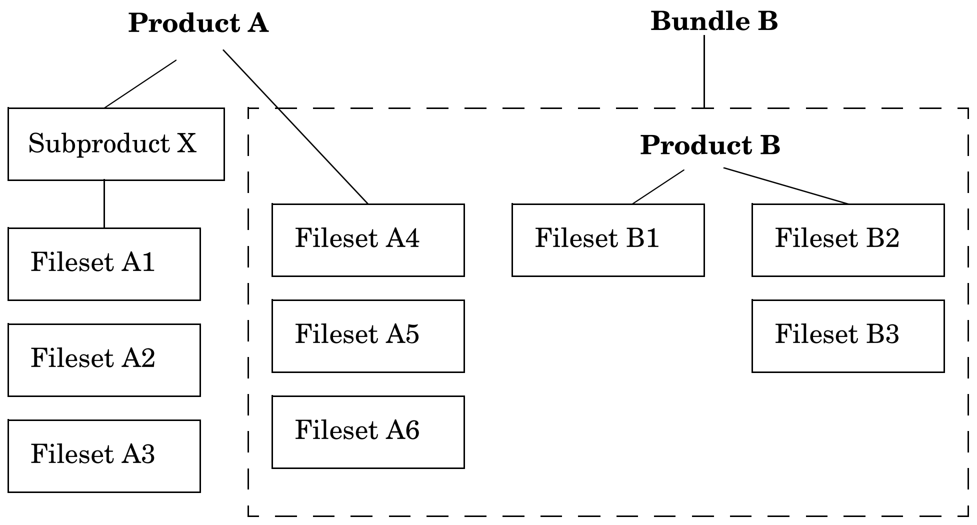
SD-UX commands work on a hierarchy of software objects that make up the applications or operating systems components you want to manage.

Software Objects

Bundles	Collections of filesets, possibly from several different products, “encapsulated” for a specific purpose. Bundles can reside in software depots, and SD-UX commands act on bundles as single entities. All HP-UX

	OS software is packaged in bundles. Bundles can consist of groups of filesets or of products. Customer creation of bundles is not supported.
Products	Collections of filesets or (optionally) subproducts and control scripts. The SD-UX commands maintain a product focus but still allow you to specify subproducts and filesets. Different versions of a product can be defined for different platforms and operating systems, as well as different revisions (releases) of the product itself. Several different versions could be included on one distribution media or depot.
Subproducts	If a product contains several filesets, subproducts can be used to group logically related filesets.
Filesets	Filesets include all the files and control scripts that make up a product. Filesets can only be part of a single product but they can be included in several different HP-UX bundles or subproducts. Like products, different versions of a fileset may be defined for different platforms and OSs. Filesets are the lowest level of object managed by SD-UX.

Figure 1-2 Example of HP-UX Software Structure



Installed Products Database

SD-UX uses the Installed Products Database (IPD) to keeps track of what software is installed on a system. The IPD is a series of files and subdirectories that contain information about all the products that are installed under the root directory (/). (For depots, this information is maintained in catalog files beneath the depot directory.)

The swinstall, swconfig, swcopy, and swremove commands automatically add to, change, and delete IPD and catalog file information as the commands are executed. The swlist and swverify commands use IPD and catalog information to affect command behavior.

The IPD keeps track of the software **state**, which includes conditions such as *installed* or *configured*.

Control Scripts

Products and filesets can contain control scripts that perform checks and other tasks not performed by SD-UX commands. SD-UX supports the following types of scripts:

Checkinstall	Analyzes each target to determine if the installation and configuration can take place. (Executed by swinstall.)
Checkremove	Analyzes each target to determine if removal and unconfiguration can take place. (Executed by swremove.)
Configure	Configures installed filesets or products. (Executed by swconfig and swinstall.)
Fix	Corrects and reports on problems in installed software. (Executed by swverify.)
Postinstall	Performs additional install operations immediately after a fileset or product has been installed. (Executed by swinstall.)
Postremove	Performs additional remove operations immediately after a fileset or product has been removed. (Executed by swremove.)
Preinstall	Performs file operations (such as removing obsolete files) immediately before installation of software files. (Executed by swinstall.)

Preremove	Performs additional file operations (such as removing files created by a preinstall script) immediately before removal of software files. (Executed by swremove.)
Request	Requests an interactive response from the user as part of the installation or configuration process. (Executed by swask, swconfig, and swinstall.)
Unconfigure	Undoes configurations performed by configure scripts. (Executed by swconfig and swremove.)
Unpostinstall	Undoes operations performed by a postinstall script in case swinstall must initiate recovery during the installation process. (Executed by swinstall.)
Unpreinstall	Undoes operations performed by a preinstall script in case SD must initiate recovery during the install process. (Executed by swinstall.)
Verify	Verifies the configuration of filesets or products (in addition to the standard swverify checks.) (Executed by swverify.)

For More Information

See Chapter 11, “Using Control Scripts,” on page 369.

Environment Variables

SD-UX commands and programs are affected by external environment variables (such as language and charset variables) and variables for use by control scripts. For a description of external environment variables, see Chapter 11, “Using Control Scripts,” on page 369.

Software Dependencies

Software that depends on other software to install or run correctly is considered to have a dependency. When you specify software for the swconfig, swcopy, swinstall, swremove, swverify commands, these commands may automatically select additional software to meet dependencies.

How Commands and Options Interact with Dependencies

Command options let you control how software dependencies are handled. For example, dependency handling in swinstall and swcopy is affected by the `enforce_dependencies` command option.

Another option that regulates dependencies is the `autoselect_dependencies` option. This option determines if the system should automatically mark software for installation or copying based on whether it meets dependencies. (See “Using Command Options” on page 59 for more information on options.)

How Dependencies Are Resolved

For a dependency to be resolved with respect to other software on the source depot it must be:

- Complete (if the dependency is an entire product or subproduct it must exist completely in the source depot)
- In the proper software state on the source (that is, available)
- Free of errors (for example, no incompatibility errors)

If the dependency is *not* available from the source during a swconfig, swcopy, swinstall, or swverify operation, the dependency must:

- Exist on the target host
- Be complete (if the dependency is an entire product or subproduct it must be completely installed)
- Be in the proper software state (the dependency must be configured if the software dependent on it is to be installed and configured, installed if software dependent on it is to be installed but not configured, or available if the software dependent on it is to be copied)
- Be free of errors (for example, no incompatibility errors).

If you select software that has a dependency and more than one available object resolves that dependency, SD-UX automatically selects the latest compatible version.

Types of Dependencies

Software packagers can define corequisites, prerequisites, and exrequisites as dependencies. These dependencies can be specified between filesets within a product, including expressions of which versions of the fileset meet the dependency. Dependencies can also be specified between a fileset and another product. Expressions for revisions and other product attributes are supported.

Corequisites	An object requires another to operate correctly, but does not imply any load order.
Prerequisites	An object requires another to be installed and/or configured correctly before it can be installed or configured respectively. Prerequisites *do* control the order of operations.
Exrequisite	An object requires the absence of another object before it can be installed or configured.

Working with Protected Software

Some HP software products are protected software. That is, you cannot install or copy the software unless you provide a codeword and customer ID. The customer ID uniquely identifies the owner of the codeword and lets you restrict installation to a specific owner. To find your codeword and customer ID, examine the CD certificate shipped with your software.

It is your responsibility to ensure that the codeword and software are used properly in this manner.

One codeword unlocks most or all of the products on your media. When you purchase additional protected products, HP provides additional codewords. SD-UX keeps tracks of codewords as you enter them. This means you do not have to enter the codeword each time you access the software.

The swinstall, swcopy, and swlist commands make use of codewords in managing software.

Using the GUI and TUI Commands

The swinstall, swcopy, swlist, swremove commands each provide a Graphical User Interface and Terminal User Interface. Advantages of the GUI/TUI include:

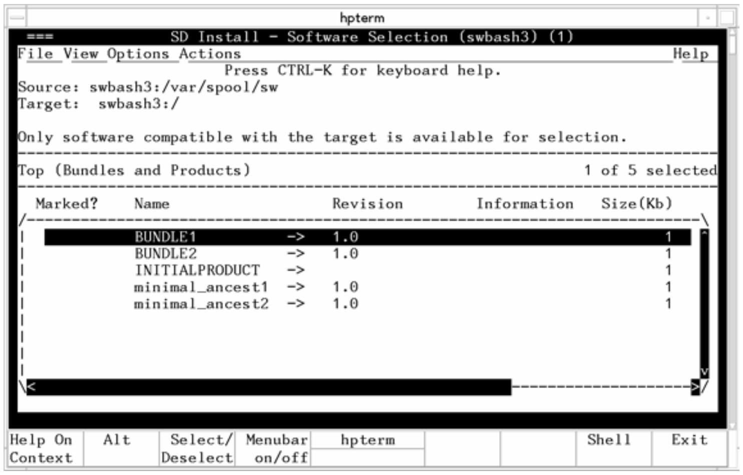
- You can quickly create and visually monitor software management tasks interactively
- You can easily analyze the effects of tasks and retry tasks that fail.
- You do not have to be familiar with a broad range of defaults, options, software selections, and other variables that are required to enter complex commands on the command line.

(Additional GUI interfaces are available if you have enabled remote operations. See Chapter 6, "Remote Operations Overview," on page 189.)

The Terminal User Interface

The terminal user interface lets you use the SD-UX GUI capabilities on systems with text-based terminals. With the TUI, you use the **Arrow**, **Tab**, **Space**, and **Return** keys to navigate.

Figure 1-3 **The Terminal User Interface (TUI)**



```
hpterm
 ===          SD Install - Software Selection (swbash3) (1)
File View Options Actions                                      Help
                      Press CTRL-K for keyboard help.
Source: swbash3:/var/spool/sw
Target:  swbash3:/

Only software compatible with the target is available for selection.
--------------------------------------------------------------------------
Top (Bundles and Products)                                1 of 5 selected
--------------------------------------------------------------------------
  Marked?    Name                Revision        Information   Size(Kb)
/------------------------------------------------------------------------\
|            BUNDLE1          ->  1.0                                1
|            BUNDLE2          ->  1.0                                1
|            INITIALPRODUCT   ->                                     1
|            minimal_ancest1  ->  1.0                                1
|            minimal_ancest2  ->  1.0                                1
|
|
|
|
\<-----------------------------------------------------------------------/
Help On   Alt    Select/  Menubar  hpterm                    Shell    Exit
Context          Deselect  on/off
```

NOTE All examples for GUI commands in this manual also apply to the TUI.

Starting the GUI/TUI Commands

To start the GUI or TUI for swinstall, swcopy, or swremove, enter:

```
/usr/sbin/swinstall
```

—or—

```
/usr/sbin/swcopy
```

—or—

```
/usr/sbin/swremove
```

TIP Put `/usr/sbin` in your PATH to avoid typing the **`/usr/sbin`** prefix.

The TUI starts by default if you have not set the DISPLAY variable.

To invoke the GUI and specify other command-line arguments at the same time, you must include the `-i` option. For example:

```
swinstall -i -s sw_server cc pascal
```

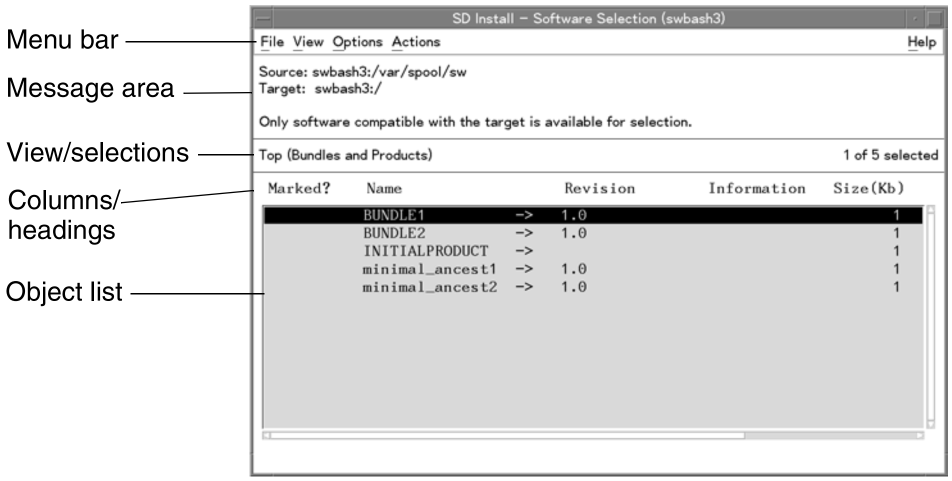
To invoke the swlist GUI, you must use always use the `swlist -i` option.

NOTE You can also launch the SD-UX GUIs from HP's ServiceControl Manager (SCM) or Systems Administration Manager (SAM) applications.

Window Components

The main GUI/TUI windows (Figure 1-4, “GUI Window Components,”) contain the following components:

Figure 1-4 **GUI Window Components**



```
Source: swbash3:/var/spool/sw
Target:  swbash3:/

Only software compatible with the target is available for selection.

Top (Bundles and Products)                                    1 of 5 selected

Marked?    Name                  Revision        Information      Size(Kb)
           BUNDLE1           ->  1.0                                     1
           BUNDLE2           ->  1.0                                     1
           INITIALPRODUCT    ->                                          1
           minimal_ancest1   ->  1.0                                     1
           minimal_ancest2   ->  1.0                                     1
```

Menu bar	Provides pull-down menus for **File**, **View**, **Options**, **Actions**, and **Help**. Each choice has additional submenus for more activities. Items in the menus may or may not appear, depending on whether selections are highlighted or not. Some actions may also be grayed out to show they are not available for a specific item.
Message area	Provides messages and system information.
View/selections	Describes the current software view and the number of items selected in the object list.
Columns	Headings for columns of information in the object list.
Object list	Lists software selections, bundles, products, targets, or other information regarding selections, analysis and details.

Opening and closing items in the object list

The Software Selection window object list is hierarchical: you can open each object in the list and show its contents. Objects in the list that contain other objects that can be opened have an arrow (→) after the name.

- To open a subproduct, double click on it, or highlight the name and then select **Actions→Open Item**. For example, to see the subproducts in the `SD-DATABASE` product, open `SD-DATABASE` by double clicking on it. The object list then displays the subproducts for `SD-DATABASE`.
- To close an object and return to the previous list, double click on the first item in the list (**..(go up)**) or highlight the item and select **Actions→Close Level**.

When a product is opened, subproducts and filesets may appear in the same list. Only products are listed together at the product level.

Filesets are the lowest level of hierarchical objects managed by SD-UX. You can not view the contents of files, but you can view the list of files in each fileset and information about each file.

Marking Items in the Object List

There are two ways to mark an object in the object list:

- Use the menu bar:
    1. Click on the object to highlight it.
    2. Select **Actions→Mark for Install** (or **Mark for Copy** or **Mark for Remove**)
- Use the pop-up menu:
    1. Click on the object to highlight it.
    2. Right click to display the pop-up menu.
    3. Select **Mark for Install** (or **Mark for Copy** or **Mark for Remove**)

Flags (`Yes`, `Partial` or blank) show whether items in the list have been marked for an activity (see the `Marked?` column).

(For the TUI, mark items by pressing **Space** when the cursor is on the item and then press the **m** key. Unmark items with the **u** key.)

Preselecting Host Files

The `defaults.hosts` file contains lists of hosts that are used by the GUI/TUI programs. This lets you use preselected choices for source and target systems. These lists are stored in the `$HOME/.swdefaults.hosts` or `/var/adm/defaults.hosts` files.

For each interactive command, target hosts containing roots or depots are specified in this file by separate lists (`hosts`, `hosts_with_depots`). The list of hosts are enclosed in {} braces and separated by white space (blank, tab and newline). For example:

```
swinstall.hosts={hostA hostB hostC hostD hostE hostF}
swcopy.hosts_with_depots={hostS}
```

When you use the program, dialog boxes that let you choose a source system from a list will display all hosts specified in `defaults.hosts` or remembered from a previous session. Once a source is successfully accessed, that host is automatically added to the list in the `defaults.hosts` file and displayed in the dialog.

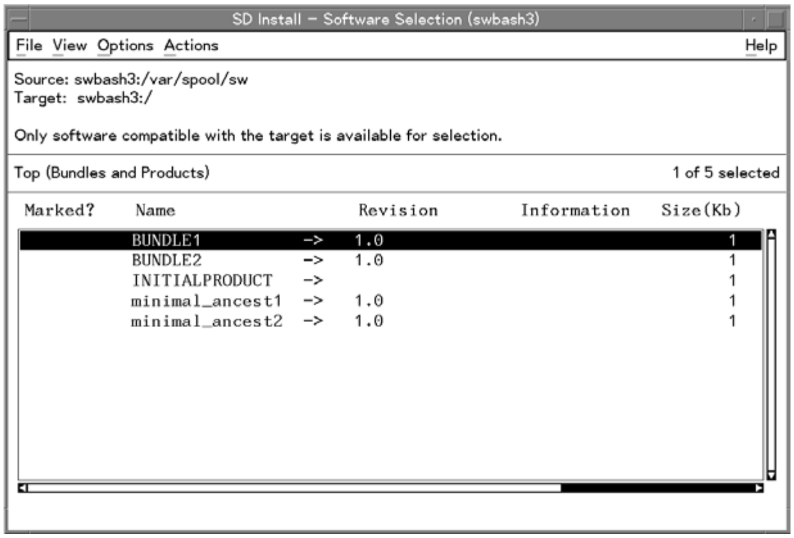
If there are no hosts specified in `defaults.hosts`, only the local host and default source host appear in the lists.

If a host system does not appear in the list, you can enter a new name from the GUI/TUI program.

Software Selection Window

The Software Selection Window (Figure 1-5, “Software Selection Window,”) is the standard window for all SD-UX GUI programs. It features the standard menu bar, message area, and object list of software available for selection. Menu items are discussed in the following sections.

Figure 1-5 Software Selection Window


SD Install – Software Selection (swbash3)
File View Options Actions
Help

```
Source: swbash3:/var/spool/sw
Target:  swbash3:/

Only software compatible with the target is available for selection.

Top (Bundles and Products)                                1 of 5 selected

Marked?    Name                 Revision     Information    Size(Kb)
           BUNDLE1          ->  1.0                                1
           BUNDLE2          ->  1.0                                1
           INITIALPRODUCT   ->                                     1
           minimal_ancest1  ->  1.0                                1
           minimal_ancest2  ->  1.0                                1
```

Session and File Management—The File Menu

The **File** menu is the primary tool for managing session files, searching, and printing.

GUI Session Files

Each invocation of one of the GUI commands defines a session. All session information—including the options used to invoke the command, source specifications, software selections, and target hosts—are automatically saved. This lets you re-execute the command even if the session ends before proper completion. (See “Session Files” on page 61.)

You can save session information into a file at any time by selecting the **Save Session** or **Save Session As** choice from the **File** menu. The **Recall Session** choice lets you import the settings from a previously saved session file. **Clear Session** resets all options and operands to their default values.

Each session is saved to a file for that command. For example:

```
$HOME/.sw/sessions/swinstall.last
$HOME/.sw/sessions/swcopy.last
$HOME/.sw/sessions/swremove.last
```

This file is overwritten by each time you start the GUI.

NOTE When you re-execute a session file, the values in the session file take precedence over values in the system defaults file. Likewise, any command line options or parameters that you specify when you invoke the GUI take precedence over the values in the session file.

Performing Text Searches

The **Search...** choice lets you perform a text search of the active list in a window.

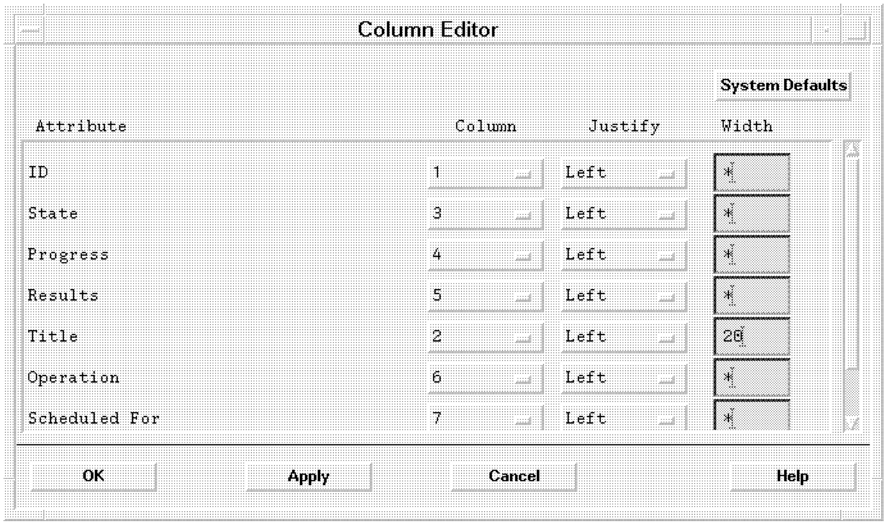
Changing Software Views—The View Menu

The **View** menu manages your window view preferences.

Columns...

The **View→Columns...** choice brings up the Column Editor dialog (Figure 1-6, “Column Editor,”), which lets you reformat the columns for the current object list. All viewable object attributes are listed.

Figure 1-6 **Column Editor**



Attribute	Column	Justify	Width
ID	1	Left	*
State	3	Left	*
Progress	4	Left	*
Results	5	Left	*
Title	2	Left	20
Operation	6	Left	*
Scheduled For	7	Left	*

The editor displays values 1 through the total number of attributes, plus an **Ignore** option, which removes that attribute from display in the object list.

You can specify an attribute’s justification by clicking on the **Left** or **Right** button in the `Justify` column.

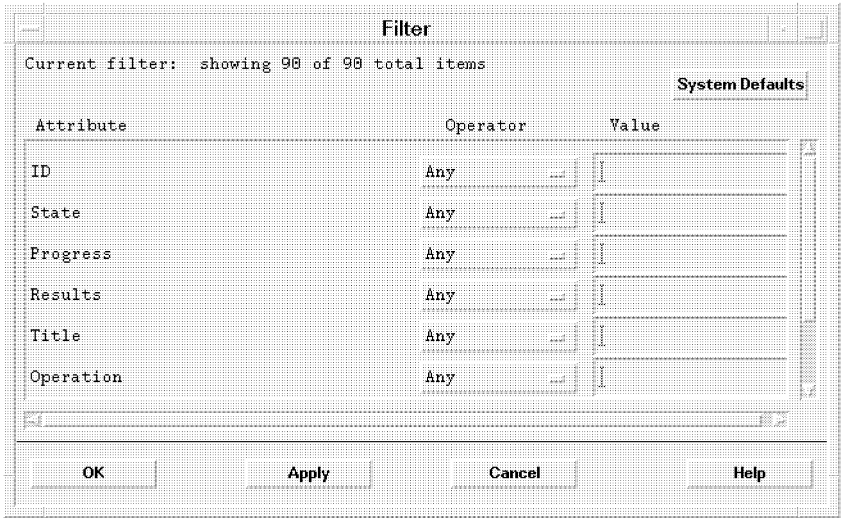
Set the column width by placing the cursor in the appropriate text field in the `Width` column, then entering the width (number of characters). Use an asterisk (*) to size the column automatically.

- To apply the changes made to the object list, select **Apply**. The list is updated to reflect any changes made, and the Column Editor dialog remains open.
- To apply the changes and close the editor, select **OK**.

- To return to the original default values, select **System Defaults**.
- To cancel any changes and return to the object list window, select **Cancel**.
- To save the changes made for the next invocation of the application, choose **View**→**Save View as Default**.

Filter...

Figure 1-7 **Filter Dialog**


Filter

```
Current filter:  showing 90 of 90 total items
```

Attribute	Operator	Value
ID	Any	
State	Any	
Progress	Any	
Results	Any	
Title	Any	
Operation	Any	

The **View**→**Filter...** choice displays the Filter dialog (Figure 1-7, "Filter Dialog,"), which lets you specify the type of filtering desired for each attribute.

The **Operator** menu button lets you specify the operator for a given attribute. The following table presents the operator types:

Table 1-3 Operator Types

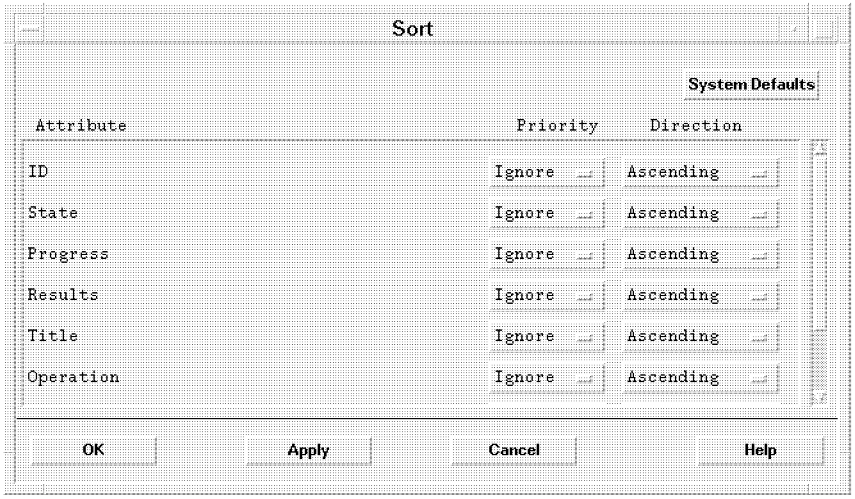
`Any`	Displays objects regardless of the value of the attribute.
`Matches`	Displays objects if their attribute value exactly matches the value specified in the `Value` column.
`Not`	Displays objects whose attribute value does not match the value specified in the `Value` column.
`Less Than`	Displays objects if their attribute value is less than the value specified in the `Value` column. *Less than* is defined as a lesser integer value or earlier in the alphabet.
`Greater Than`	Displays objects if their attribute value is greater than the value specified in the `Value` column. *Greater than* is defined as a higher integer value or later in the alphabet.

- For `Matches` and `Not`, use an asterisk (*)as a wildcard, and a question mark (?) to match any single character.
- Select **Apply** to apply the changes made to the object list and leave the Filter dialog open.
- Select **OK** to apply the changes and close the Filter dialog.
- To return to the original default values, select **System Defaults**.
- Select **Cancel** to ignore any changes made and close the Filter dialog.
- To save the changes made for the next invocation of the application, choose **View→Save View as Default**.

Sort...

The **View→Sort...** choice displays the Sort dialog (Figure 1-8, “Sort Dialog,”), which lets you specify a sort method for the object list. All viewable object attributes are listed. For each attribute, you can specify the type of sort desired.

Figure 1-8 **Sort Dialog**



Attribute	Priority	Direction
ID	Ignore	Ascending
State	Ignore	Ascending
Progress	Ignore	Ascending
Results	Ignore	Ascending
Title	Ignore	Ascending
Operation	Ignore	Ascending

The `Priority` column displays values 1 through the total number of attributes, plus an `Ignore` option, which excludes the attribute from the sort. A sort priority of 1 sorts the list first on that attribute.

- To specify whether the sort is ascending or descending, select the `Direction` menu button.
- To apply the changes to the objects list and remain in the Sort dialog, select **Apply**.
- Select **OK** to apply the sort and close the Sort dialog.
- Select **Cancel** to ignore any changes made and close the Sort dialog.
- To return to the original default sort values, select **System Defaults**.

Save View as Default

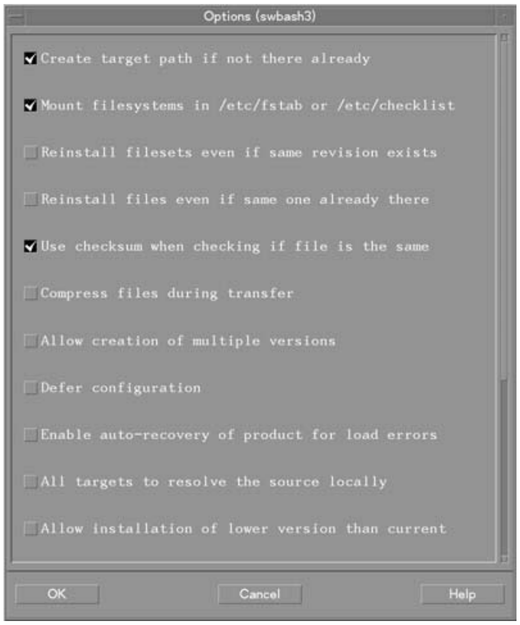
To save any changes for future sessions, choose **View→Save View as Default**. Any changes you made to your view preferences are saved in the following file, in which `username` is your log-in name: `/var/adm/sw/ui/preferences/username.prefs`.

Changing Options and Refreshing the Object List—The Options Menu

The **Options** menu lets you refresh the object list and change the default values of options that control command behaviors and policies. Selecting **Options→Refresh List** updates the object list to reflect any changes.

Selecting **Options→Change Options** opens the Options dialog (Figure 1-9, "Options Editor Dialog,"), which lets you change a limited set of options for the command. These options are changed only for the duration of the interactive session. To change options for subsequent sessions, you must save a session file (see "Session and File Management—The File Menu" on page 41) or edit one of the options files (see "Using Command Options" on page 59).

Figure 1-9 **Options Editor Dialog**



- [✓] Create target path if not there already
- [✓] Mount filesystems in /etc/fstab or /etc/checklist
- [ ] Reinstall filesets even if same revision exists
- [ ] Reinstall files even if same one already there
- [✓] Use checksum when checking if file is the same
- [ ] Compress files during transfer
- [ ] Allow creation of multiple versions
- [ ] Defer configuration
- [ ] Enable auto-recovery of product for load errors
- [ ] All targets to resolve the source locally
- [ ] Allow installation of lower version than current

NOTE Use caution when changing option values. They allow useful flexibility but can produce harmful results if changed to an inappropriate value. Use the online help and consult Appendix A, “Command Options,” on page 421 to understand fully each option before you change it.

Performing Actions—The Actions Menu

Each **Action** menu in the GUI/TUI programs has series of actions for that command. These actions vary according to which command you invoke. (You may have to click on an item in the object list to enable some of the actions that are grayed out.) The following actions are common to swinstall, swcopy, swlist, and swremove.

Open Item/Close Level

The **Open Item** or **Close Level** menu choices let you see the contents of a selected object or close it.

Each object list is hierarchical. Objects that have an arrow (→) after the name can be opened to reveal other items. For example, to see the subproducts in a particular product, you can open that product by double clicking on the object or by selecting **Actions ->Open Item**. The object list then shows a listing of the subproducts for that product. If you want to open the subproduct, double click on it and its filesets are displayed. (In the TUI, move the cursor to the item you want to open and click **Return**.)

When the product is opened, all of its subproducts (and filesets that are not part of a subproduct) are shown in the list. At the product level, only products are listed together. If the software view is Bundle and the bundle is opened, all HP-UX OS products that are wholly or partially contained in the bundle will be shown. When one of the products is opened, only subproducts and filesets in the open product and open bundle are shown.

To close an object and return to the previous list, double click on the first item in the list (**. .(go up)**) or select **Actions->Close Level**. (In the TUI, you must use Close Level in the Actions menu or press **Return** while highlighting the (**. .(go up)**) item.)

Add/Save Software Group

These choices let you save and re-use groups of marked software.

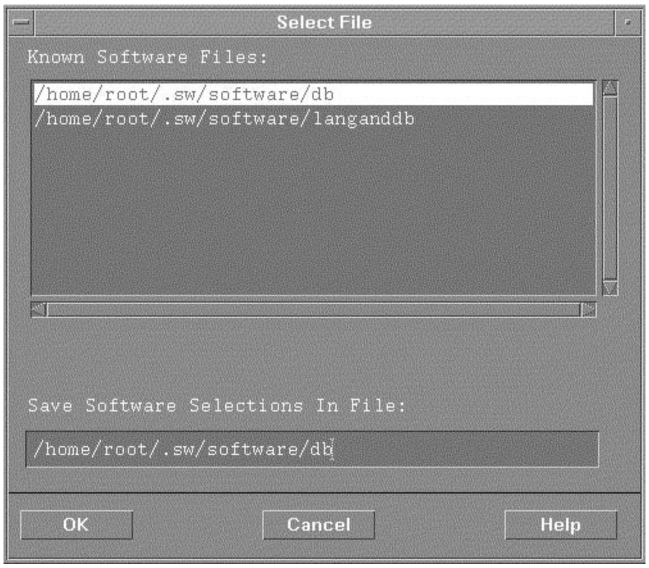
The **Save Software Group** menu choice opens the Save Software Group dialog (Figure 1-10, “Save Software Group Dialog,”), which saves the current list of marked software as a group. SD stores the group definition in `$HOME/.sw/software/` or a directory you specify.

You can recall and re-use a previously saved group of software selections by using the **Add Software Group** menu choice.

NOTE Software automatically marked due to dependencies is not included in a software group. Dependencies are recomputed each time you select **Add Software Group**. See “Software Dependencies” on page 33 for more information about dependencies

---

Figure 1-10 **Save Software Group Dialog**



Known Software Files:

```
/home/root/.sw/software/db
/home/root/.sw/software/langanddb
```

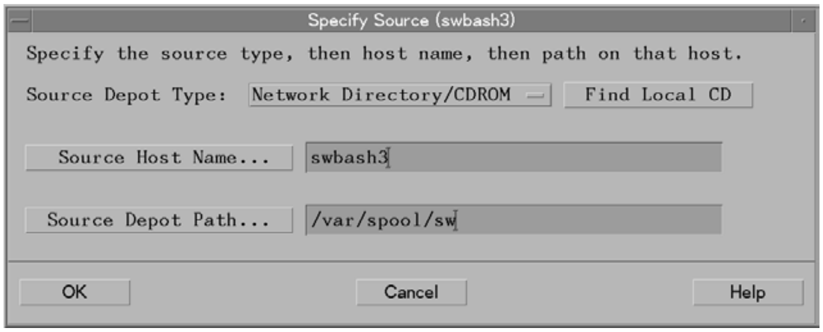
Save Software Selections In File:

```
/home/root/.sw/software/db
```

Change Source

The **Change Source...** menu choice opens the Change Source dialog (Figure 1-11, "Change Source Dialog,"), which lets you change the source for the software to be used. The **Root Path** button opens a list of target paths from which to select (Figure 1-13, "Root Path Dialog,").

Figure 1-11 **Change Source Dialog**



1. (Optional) To specify another host system, type a source host name, or:
   a. Click on the **Source Host Name** button. The system displays a dialog that lists all host system names contained in the `defaults.hosts` file (`$HOME/.sw/defaults.hosts` or `/var/adm/sw/defaults.hosts`).
   b. Choose a host name from the list.
   c. Click **OK**. The host name appears in the appropriate box in the Specify Source dialog.
2. (Optional) To specify the path to the depot, type a new path, or:
   a. Click on the **Source Depot Path** button to display a list of registered depots on the source host.
   b. Highlight one of the depots.
   c. Click **OK** to make it appear in the Specify Source dialog.
3. Click **OK**. The Specify Source dialog closes, and the Software Selection window displays the software contained in the depot you specified.

Change Target

The **Change Target...** menu choice opens the Change Target dialog (Figure 1-12, “Change Target Dialog,”), which lets you change the targets of your software operation. The **Root Path** button opens a list of target paths from which to select (Figure 1-13, “Root Path Dialog,”).

For SD-UX local operations, the target is always a directory on the local host. See Chapter 6, “Remote Operations Overview,” on page 189 for information about specifying remote targets.

Figure 1-12 Change Target Dialog

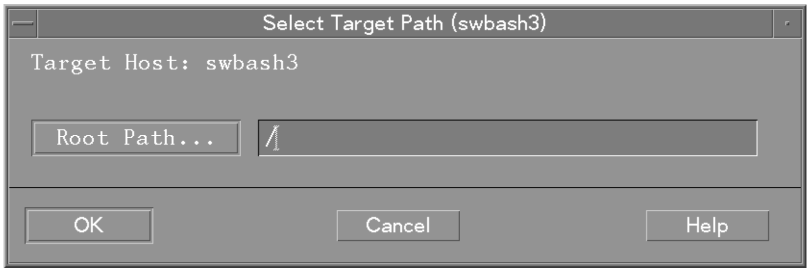
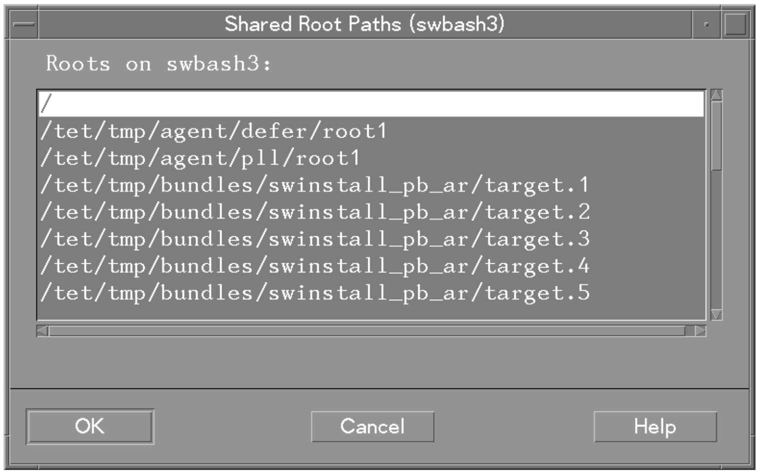


Figure 1-13 Root Path Dialog

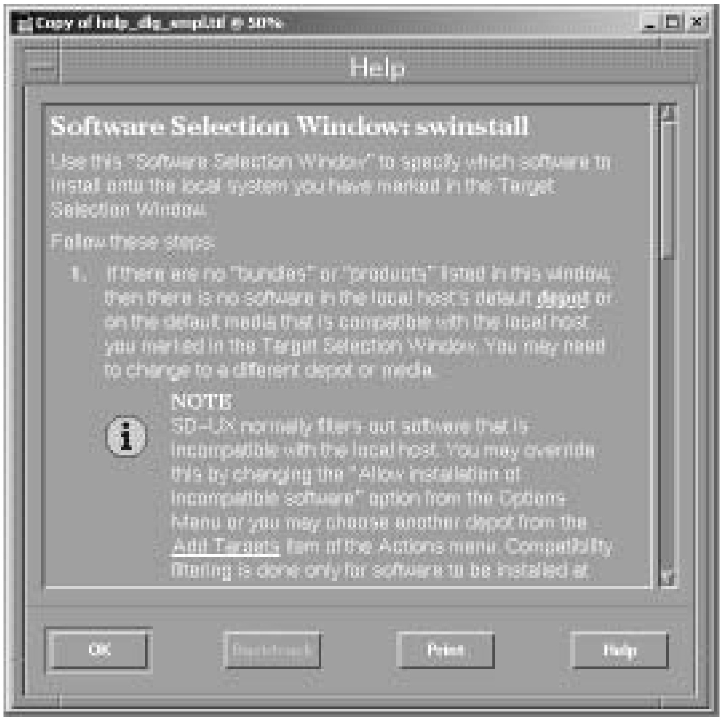


```
/
/tet/tmp/agent/defer/root1
/tet/tmp/agent/pll/root1
/tet/tmp/bundles/swinstall_pb_ar/target.1
/tet/tmp/bundles/swinstall_pb_ar/target.2
/tet/tmp/bundles/swinstall_pb_ar/target.3
/tet/tmp/bundles/swinstall_pb_ar/target.4
/tet/tmp/bundles/swinstall_pb_ar/target.5
```

Getting Help—The Help Menu

All the GUI and TUI programs have an on-line help system. Each screen, dialog, or menu choice has associated help instructions that explain the activity.

Figure 1-14 **Typical On-Line Help Screen**



Software Selection Window: swinstall

Use this "Software Selection Window" to specify which software to install onto the local system you have marked in the Target Selection Window.

Follow these steps:

1. If there are no "bundles" or "products" listed in this window, then there is no software in the local host's default depot or on the default media that is compatible with the local host you marked in the Target Selection Window. You may need to change to a different depot or media.

NOTE
SD-UX normally filters out software that is incompatible with the local host. You may override this by changing the "Allow installation of incompatible software" option from the Options Menu or you may choose another depot from the Add Targets item of the Actions menu. Compatibility filtering is done only for software to be installed at

To get "context-sensitive" help for individual menu choices, fields, options, or buttons on the various windows and menus, place the cursor on an item and press the **F1** key on your keyboard (**Ctrl-F** in the TUI). This displays specific help for that item.

To view overview information for each major screen, to get help on keyboard usage, or to view other product information, select the **Help** menu from in the menu bar.

Overview...

This menu item provides information about the currently active SD-UX screen. This includes a list of the tasks you can do in that screen and a short description of the different areas of the screen and links to related topics.

Keyboard...

This menu item brings up help on how to use the keyboard to control the application, covering topics such as selection, menu bar activation and traversal, dialog box traversal, etc.

Using Help...

This menu item displays information about how to use the Help system.

Product Information...

This menu item displays copyright and revision information for SD-UX.

XToolkit Options and Changing Display Fonts

The GUI commands support the following subset of the HP-UX XToolkit command line options:

- `-bg` or `-background`
- `-fg` or `-foreground`
- `-display`
- `-name`
- `-xrm`

Note that the SD-UX commands do not support the XToolkit `-fn` or the `-font` option used to change display fonts.

SD-UX commands do, however, recognize most Motif™ standard resources when running in the X11/Motif environment, plus the following additional resources:

`*systemFont`	Specifies the variable-width font used in the GUI menu bars and other areas where a variable width font is applicable. The default size is 8x13.

`*userFont`	Specifies the fixed-width font used in all other GUI displays. This font should be the same basic size as the `*systemFont` only in the fixed width style. The default size is also 8x13.

Here is an example of how to change the size of your fixed width font from 8x13 to 6x13:

```
swinstall -xrm ’Swinstall*userFont: user6x13’
```

Here is how to change the variable width font style to 12 point HP Roman 8:

```
swinstall -xrm ’Swinstall*systemFont: \
-adobe-courier-medium-r-normal12-120-75-75-m-70-hp-\
roman8’
```

You can also modify the defaults file (in `/usr/lib/X11/app-defaults`) for each command with a Graphical User Interface so that a resource will be set each time you invoke a specific command. Here is an example of an app-defaults file for swremove:

`swremove app-defaults`

```
Swremove*foreground:  red
Swremove*background:  white
Swremove*userFont:    hp8.8x16b
Swremove*systemFont:  -adobe-courier-medium-r-normal12-120-75-75-m
-70-hp-roman8
```

Working from the Command Line

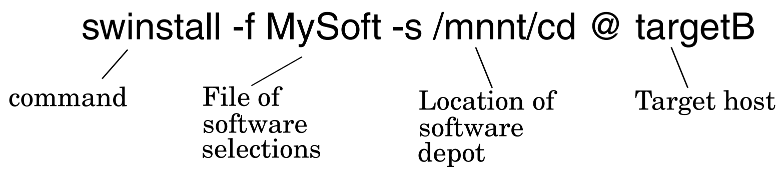
You can invoke all SD-UX commands non-interactively via the command line. This section provides reference information about command-line features available across most of the commands.

The command line is most effective for:

- Quickly executing simple commands
- Executing tasks that take a long time to accomplish
- Creating commands for later execution by scripts

A typical command line might look like this:

Figure 1-15 **Sample Command**


swinstall -f MySoft -s /mnnt/cd @ targetB
command
File of software selections
Location of software depot
Target host

The example shows that you have several ways to specify SD-UX behavior including **command-line options** (such as `-f` and `-s`), **input files** (`mysoft` and `/mnt/cd`), and **target selections**.

A complete list of command line components includes:

- Software selections and software selection files (page 56)
- Target selections and target selection files (page 58)
- Command-line options (page 59)
- Session files (page 61)

Each item on this list is discussed in more detail in the following sections.

Software Selections

Software selections let you specify software in great detail. You can also use an input file to specify software.

Syntax

The *`software_selections`* syntax is identical for all SD-UX commands that require it:

bundle[.*product*[.*subproduct*][.*fileset*]][,*version*]
product[.*subproduct*][.*fileset*][,*version*]

- The = (equals) relational operator lets you specify selections with the following shell wildcard and pattern-matching notations:

[ ]	Square brackets—groups an expression
*	Asterisk—wildcard for multiple characters
?	Question mark—wildcard for a single character

  For example, the following expression installs all bundles and products with tags that end with `man`:

  ```
  swinstall -s sw_server \*man
  ```

- Bundles and products are recursive. Bundles can contain other bundles. For example:

  ```
  swinstall bun1.bun2.prod.sub1.fset,r=1.0
  ```

  or (using expressions):

  ```
  swinstall bun[12].bun?.prod.sub*,a=HP-UX
  ```

- The `\*` software specification selects all products.

CAUTION To avoid data loss, use the `\*` specification with considerable care (such as when removing software from the root directory, `/`).

The version component has the form:

`[,r <`*`op`*`> `*`revision`*`][,a <`*`op`*`> `*`arch`*`][,v <`*`op`*`> `*`vendor`*`]`
`[,c <`*`op`*`> `*`category`*`][,q=`*`qualifier`*`][,l=`*`location`*`]`
`[,fr <`*`op`*`> `*`revision`*`][,fa <`*`op`*`> `*`arch`*`]`

where:

- Fully qualified software specifications include the `r=`, `a=`, and `v=` version components, even if they contain empty strings. For installed software, `l=` is also required.
- All version components are repeatable within a single specification (e.g. **`r>=A.12`**, **`r<A.20`**). If multiple components are used, the selection must match all components.
- The <*op*> (relational operator) component performs individual comparisons on dot-separated fields and can be of the form:

  =, ==, >=, <=, <, >, or `!=`

  For example, **`r>=B.11.11`** chooses all revisions greater than or equal to `B.11.11`. The system compares each dot-separated field to find matches.
- The = (equals) relational operator lets you specify selections with the shell wildcard and pattern-matching notations: `[ ]`, `*`, `?`, and `!`

  For example, the expression **`r=1[01].*`** returns any revision in version 10 or version 11.
- No space or tab characters are allowed in a software selection.
- *`qualifier`* is a string that can be attached to any product or bundle to help you filter a software specification.
- *`location`* applies only to installed software and refers to software installed to a location other than the default product directory.
- `fr` and `fa` apply only to filesets.
- A software *`instance_id`* can take the place of the version component. It has the form:

  `[`*`instance_id`*`]`

  within the context of an exported catalog, where instance_id is an integer that distinguishes versions of products and bundles with the same tag.

Software Files

To keep the command line shorter, software selection input files let you specify long lists of software products. With a software selection file, you only have to specify the single file name.

The `-f` command-line option lets you specify a software selection file. For example:

```
swinstall -f mysoft -s /mnt/cd @ targetB
```

In this example, the file `mysoft` (which resides in the current working directory for software files) contains a list of software selections for the depot /mnt/cd.

In the software file, blank lines and comments (lines beginning with #) are ignored. Each software selection must be specified on a separate line.

Target Selections

Target selections follow software and source depot selections. If no target selection is named, the target on which the operation will be performed is assumed to be the root (`/`) directory on your local host. So, you do not have to use the @ sign and [*host*][:][*/directory*] designation (described below) if you are operating on the local host or default depot directory.

Syntax

The *target_selections* syntax is identical for all SD-UX commands that require it:

@ [*host*][:][/*directory*]

- The @ character is optional if you are using the local host and default directory. If it is used, it acts as a separator between operands and the destination.
- Only one @ character is needed.
- You can specify the host by its host name, domain name, or internet address. A directory must be specified by an absolute path.
- The `:` (colon) is required if you specify both a host and directory.
- On some systems, the @ character is used as the kill function. Type `stty` on your system to see if the @ character is mapped to any other function on your system. If it is, remove the mapping, change the mapping, or use \@.

Target Files

To keep the command line shorter, target selection input files let you specify long lists of targets. With a target selection file, you only have to specify the single file name.

The `-t` command-line option lets you specify a target file. For example:

```
swinstall -f mysoft -s /mnt/cd -t mytargs
```

In this example, the file **`mytargs`** (which resides in the current working directory) contains a list of target selections for the swinstall command.

In the target file, blank lines and comments (lines beginning with #) are ignored. Each target selection must be specified on a separate line and must consist of a host name or network address, optionally followed by a colon and a full path: *`host`*[`:`*`/directory`*]

Using Command Options

You can control many SD-UX command policies and behaviors by setting the appropriate command options. You can change the default values of options using predefined files or values you specify directly on the command-line. Altering default values with files can help when you don't want to specify command behavior every time you invoke the command.

These rules govern the way the defaults work:

1. Options in `/var/adm/sw/defaults` affect all SD-UX commands on that system. This file can change the default behavior for all commands to which an option applies or for specific commands only.
2. Options in your personal `$HOME/.swdefaults` file affect only you and not the entire system.
3. Options read from a session file affect only that session.
4. Options changed on the command line by the `-X` *`option_file`* or the `-x` *`option=value`* arguments override the system-wide and personal options files but affect only that invocation of the command.

For system-wide policy setting, use the `/var/adm/sw/defaults` files. Keep in mind, however, that users may override these options with their own `$HOME/.swdefaults` file, session files, or command line changes.

The template file `/usr/lib/sw/sys.defaults` provides an easy way to change system-wide or personal option files.

The template file lists (as comments):

- All command options
- The commands to which each option applies
- Possible values for each option
- The resulting system behavior for each value.

You can copy values from this file into the system defaults file (`/var/adm/sw/defaults`), your personal defaults file (`$HOME/.swdefaults`), or an input file (with the `-X` *`input_file`* option) and edit them to affect SD-UX behavior.

Option files use this syntax:

[*command*.]*option*=*value*

- The optional *command* is the name of a SD-UX command. Specifying a command name changes the default behavior for that command only. A period must follow a command name.
- *option* is the name of the default option. An equals sign must follow the option name.
- *value* is one of the allowable values for that option.

NOTE You must restart the SD-UX daemon after changing swagentd options, or the daemon will not recognize the changes. To restart the daemon, type:

```
/usr/sbin/swagentd -r
```

Examples

To change the default value of `use_alternate_source` to true for all users for all future sessions for all commands to which the option applies, place the following line in the `/var/adm/sw/defaults` file:

```
use_alternate_source=true
```

To change the default value of `use_alternate_source` to false for your own invocations of the command, place the following line in your `$HOME/.swdefaults` file:

```
swinstall.use_alternate_source=false
```

To start an interactive swinstall session using the options stored in `my_install_defaults` to override any system-wide or personal defaults file values:

```
    swinstall -i -X my_install_defaults=true
```

To start an interactive install session and reset the `use_alternate_source` default for this session only:

```
    swinstall -i -x use_alternate_source
```

See Appendix A, "Command Options," on page 421 for a complete listing of defaults and their values and descriptions.

CAUTION Changing the default values for command options can cause harmful results if you specify inappropriate values.

Session Files

Before any SD-UX task starts, the system automatically saves the current command options, source information, software selections, target selections, etc., into a **session file**. You can then re-use this session information at a later time, even if the command fails.

Session information is saved in the `$HOME/.sw/sessions/` directory as *`command`*`.last` in which *`command`* is the name of the command. Each time you save a session file, it overwrites the previously stored one. (To save multiple session files, you can rename each session file after you invoke the command.)

To re-use the automatically saved session file, invoke the command with the `-S` *`swcommand`*`.last` argument. For example:

```
swinstall -S swinstall.last
```

If you want to save a session file to somewhere other than the default sessions directory, use the `-C` *`session_file`* argument and supply your own absolute path to the file you wish to save. If you do not specify a directory, the default location for the session file is `$HOME/.sw/sessions/`.

To re-execute a session from a command line, specify the session file as the argument for the `-S` *`session_file`* option.

Note that when you re-execute a session file, the session file values take precedence over values in the system defaults file or personal defaults file. Likewise, any command line options or parameters that you specify when you invoke the command take precedence over the values in the session file.

Here is a sample a session file. It uses the same syntax as the defaults files:

```
# swinstall session file
#
# Filename      /users/fred/.sw/sessions/swinstall.last
# Date saved    05/26/01 15:59:41 MDT
swinstall.allow_downdate = true
swinstall.allow_incompatible = false
swinstall.allow_multiple_versions = false
swinstall.autoreboot = false
swinstall.autorecover_product = false
swinstall.compress_files = false
swinstall.create_target_path = true
...
```

(A typical swinstall session file has approximately 70 lines.)

2 Installing Software

This chapter discusses how to use the swinstall, swconfig, and swverify commands to install, configure, and verify software.

- swinstall installs software from a depot and performs automatic configuration of software.
- swconfig lets you configure, unconfigure, or reconfigure previously installed software.
- swverify lets you check that software was installed correctly and run scripts to perform additional verification tasks or fix specific problems.

Table 2-1 Chapter Topics

Topics:
"Installation with swinstall" on page 64
"Configuring Your Installation (swconfig)" on page 82
"Verifying Your Installation (swverify)" on page 89

Installation with swinstall

The swinstall command installs software from a software source (a depot or physical media) to your local host.

Features and Limitations

- Optional GUI.
- Compatibility filtering to ensure the software will run on the installed system.
- Ability to perform kernel rebuilding or rebooting.
- Automatic use of dependencies to automatically select software on which to operate (in addition to any software you specify directly).
- Ability to run control scripts as part of the installation:

Checkinstall	Analyses each target to determine if the installation and configuration can take place.
Preinstall	Performs file operations (such as removing obsolete files) before installation of software files.
Request	Requests an interactive response from the user as part of the installation or configuration process. (Executed by swask, swconfig, and swinstall.)
Configure	Configures installed filesets or products. (See “Configuring Your Installation (swconfig)” on page 82.)
Postinstall	Performs additional install operations (such as resetting default files) immediately after a fileset or product has been installed.
Unpostinstall	Undoes a postinstall script in case swinstall must initiate recovery during the installation process.
Unpreinstall	An undo preinstall script in case SD must initiate recovery during the install process.

(See Chapter 11, “Using Control Scripts,” on page 369)

- Software can be installed to alternate root directories.

Installing with the GUI

Overview

This section provides an overview of the swinstall GUI.

- In general, all information presented in “Installing from the Command Line” on page 73 also applies to the swinstall GUI.
- This section also refers to additional information about standard GUI elements, discussed in “Using the GUI and TUI Commands” on page 35.
- All information in this section also applies to the TUI program unless otherwise noted. See “The Terminal User Interface” on page 35 for more information.

There are five steps in the GUI install process:

Table 2-2 **GUI Installation Steps**

I. Start-Up	Start the swinstall GUI.
II. Select Source	Provide the location of the software depot from which the software will be installed.
III. Select Software	Choose the software to install.
IV. Analysis (Preview)	Analyze (preview) the installation to determine if the selected software can be installed successfully.
V. Installation	Perform the actual software installation.

Step I: Start-Up

To start the GUI or TUI for an install session, type:

```
/usr/sbin/swinstall
```

The GUI is automatically invoked *unless* you also specify software on the command line. To invoke the GUI *and* specify software, include the `-i` option. For example, to use the GUI for a preview (analysis only) session with `BUNDLE1`, type:

```
swinstall -i -p /MyDepot/BUNDLE1
```

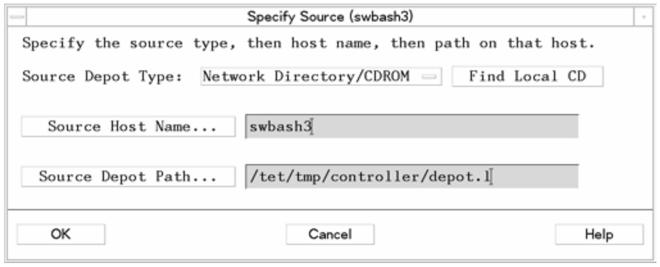
The Software Selection window appears with the Specify Source dialog superimposed over it.

Step II: Select Source

In this step, you must specify the source depot that contains the software you want to install. The Specify Source dialog (Figure 2-1, “Specify Source Dialog,”) automatically lists the local host and default depot path.

(This step is skipped if you include the `-s source` option when you invoke the GUI.)

Figure 2-1 **Specify Source Dialog**

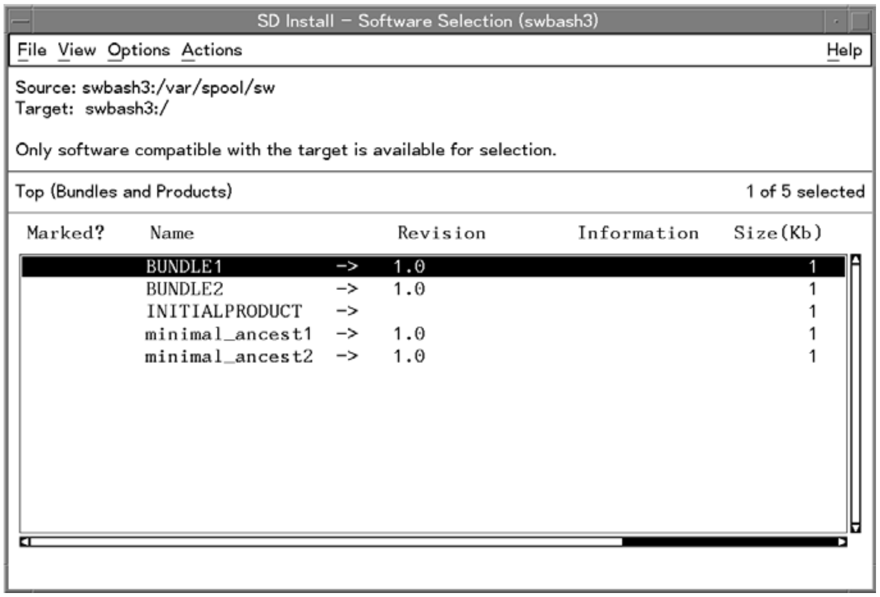


1. (Optional) To specify another host system, type a source host name, or:
   a. Click on the **Source Host Name** button. The system displays a dialog that lists all host system names contained in the `defaults.hosts` file (`$HOME/.sw/defaults.hosts` or `/var/adm/sw/defaults.hosts`).
   b. Choose a host name from the list.
   c. Click **OK**. The host name appears in the appropriate box in the Specify Source dialog.
2. (Optional) To specify the path to the depot, type a new path, or:
   a. Click on the **Source Depot Path** button to display a list of registered depots on the source host.
   b. Highlight one of the depots.
   c. Click **OK** to make it appear in the Specify Source dialog.
3. Click **OK**. The Specify Source dialog closes, and the Software Selection window displays the software contained in the depot you specified.

Step III: Select Software

In this step, you use the Software Selection window to select the software you want to install.

Figure 2-2 swinstall Software Selection Window



SD Install – Software Selection (swbash3)
File View Options Actions
Help

Source: swbash3:/var/spool/sw
Target: swbash3:/

Only software compatible with the target is available for selection.

Top (Bundles and Products) 1 of 5 selected

```
Marked?   Name                    Revision       Information    Size(Kb)
          BUNDLE1          ->     1.0                                  1
          BUNDLE2          ->     1.0                                  1
          INITIALPRODUCT   ->                                          1
          minimal_ancest1  ->     1.0                                  1
          minimal_ancest2  ->     1.0                                  1
```

1. Select software from the object list:
   a. Highlight an item
   b. Select **Actions→Mark For Install**

      — *or* —

      Right-click to display the pop-up, then select **Mark For Install**

   The `Marked?` flag in the object list changes to `Yes` to match your selection. (The flag `Partial` may appear if you select only a component of a software object or if such components are automatically selected due to dependencies.)

NOTE If multiple versions of a product exist in the same depot, SD-UX lets you select only one version during each installation session.

2. (Optional) Use choices from the **Actions** menu:
   - **Match What Target Has** examines your current Installed Product Database to match your existing filesets with new filesets (those with the same names) that you are going to install. This feature is most helpful when you are updating a system to newer versions of the same software. This option can be set from the Options Editor.
   - **Add Software Group** displays a list of previously saved software group files or lets you specify a directory. Selecting a file adds the software selections in the file to any selections you have already made in the Software Selection window.
   - **Save Software Group** lets you save your current list of marked software as a group.
   - **Manage Patch Selections** lets you select from a list of patches to install, select filters for patches, and set other patch options. (See “Installing Patches” on page 77 for more information.)
   - **Change Source...** cancels your software selections and returns you to the Specify Source dialog.
   - **Add New Codeword** lets you add a new codeword to unlock protected software. (This option is available only when SD-UX detects that the source contains protected software.)
   - **Change Target...** displays the Select Target Path dialog. This lets you specify an alternate root for products that are relocatable.
   - **Show Description of Software** (available only for a single item highlighted in the object list) displays more information on the selected software.
3. Select **Actions**→**Install** to start the analysis (preview) step. The Analysis dialog appears.

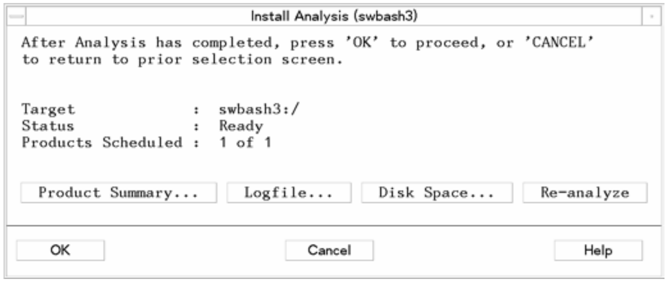
Step IV: Analysis (Preview)

In this step, SD-UX analyzes the software you have selected.

The Analysis window displays status information about the analysis process. When the analysis is complete and the host status shows Ready, click **OK** to start the actual installation (see “Step V: Installation” on page 72). The Analysis dialog is then replaced by the Install dialog.

If you started a preview session, the install stops after the analysis. Clicking **OK** returns you to the Software Selection window.

Figure 2-3 Analysis Dialog



```
After Analysis has completed, press 'OK' to proceed, or 'CANCEL'
to return to prior selection screen.


Target            :  swbash3:/
Status            :  Ready
Products Scheduled :  1 of 1
```

The following actions are available:

- **Product Summary** gives additional information about the product or bundle and provides a **Product Description** button that displays information about additional information about dependencies, copyright, vendor, etc.
- **Logfile** presents a scrollable view of detailed install information written to the logfile.
- **Disk Space** displays the Disk Space Analysis window (Figure 2-4, “Disk Space Analysis Window,”) which shows:
  - — The file system mount point,
  - — How much disk space was available before installation,
  - — How much will be available after installation,
  - — What percent of the disk’s capacity will be used.
  - — How much space must be freed to complete the operation.

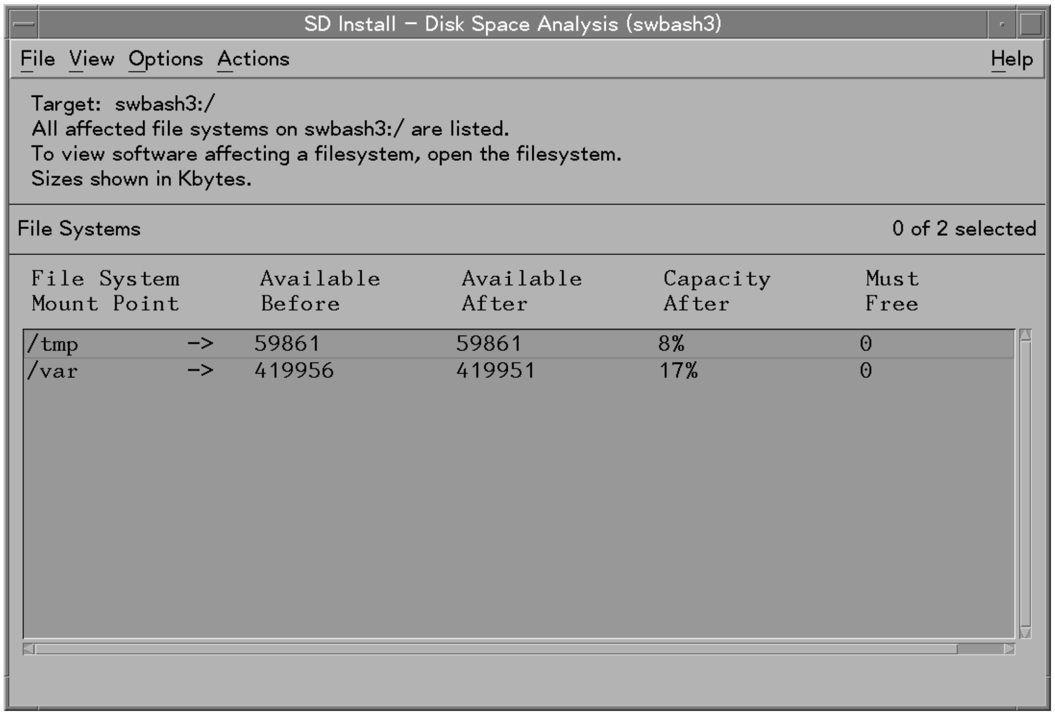
Menu choices in this window let you:

— Search the object list.

— Open items to look at the projected size requirements for specific filesets.

- **Re-analyze** repeats the analysis process.

Figure 2-4 Disk Space Analysis Window



Target: swbash3:/
All affected file systems on swbash3:/ are listed.
To view software affecting a filesystem, open the filesystem.
Sizes shown in Kbytes.

File Systems 0 of 2 selected

```
File System       Available    Available    Capacity    Must
Mount Point       Before       After        After       Free
/tmp       ->     59861        59861        8%          0
/var       ->     419956       419951       17%         0
```

When Analysis completes, the status for any host displays as either `Ready` or `Excluded from task`. If *any* of the selected software can be installed onto the host, the status shows `Ready`. If *none* of the selected software can be installed onto the host, the status shows `Excluded from task`.

The following list summarizes the status results. You can find details about most problems by clicking the **Logfile** button.

`Ready`	There were no errors or warnings during analysis. The installation may proceed without problems.
`Ready with Warnings`	Warnings were generated during the analysis. Errors and warnings are logged in the logfile.
`Ready with Errors`	At least one product selected will be installed or copied. However, one or more products selected are excluded from the task because of analysis errors. Errors and warnings are logged in the logfile.
`Communication failure`	Contact or communication with the intended target or source has been lost.
`Excluded due to errors`	Some kind of global error has occurred. For example, the system might not be able to mount the file system.
`Disk Space Failure`	The installation will exceed the space available on the intended disk storage. For details, click the **Disk Space** button.

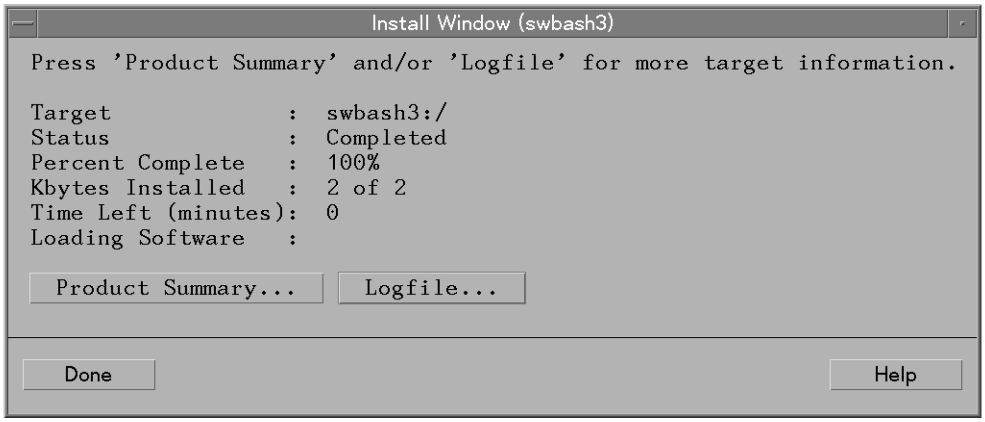
The `Products Scheduled` row shows the number of products ready for installation out of all products selected. These include:

- Products selected only because of dependencies
- Partially selected products
- Other products and bundles that were selected

Step V: Installation In this step, SD-UX proceeds with the actual installation.

After you click **OK** in the Analysis window, SD-UX starts installation and displays the Install Window, which shows status information.

Figure 2-5 **Install Window**



```
Press 'Product Summary' and/or 'Logfile' for more target information.

Target              :  swbash3:/
Status              :  Completed
Percent Complete    :  100%
Kbytes Installed    :  2 of 2
Time Left (minutes):  0
Loading Software    :
```

These action buttons are available:

- **Done** returns you to the Software Selection Window. You can then begin another install or exit the GUI (**File**→**Exit**).
- **Product Summary** display installation and product information (name, revision, installation results, installation summary).
- **Logfile** displays the logfile.
- (Appears only for kernel installations) **Resume** restarts a suspended installation. This lets you fix problems before continuing.
- (Appears only for kernel installations) **Abort** cancels a suspended installation.

Installation may suspend if:

- File loading fails
- An error occurs in a script
- Customization for kernel-related filesets fails
- A kernel build fails
- A tape change is needed (if you are installing from multi-tape media)

Installing from the Command Line

Swinstall syntax

The syntax for swinstall is:

`swinstall` [*`XToolkit Options`*] [`-i`] [`-p`] [`-r`] [`-v`] [`-c`*`catalog`*]
[`-C` *`session_file`*] [`-f` *`software_file`*] [`-Q` *`date`*] [`-s` *`source`*]
[`-S` *`session_file`*] [`-t` *`target_file`*] [`-x` *`option=value`*]
[`-X` *`option_file`*] [*`software_selections`*] [`@` *`target_selections`*]

Options and Operands

XToolkit Options	X window options for the GUI. See “XToolkit Options and Changing Display Fonts” on page 53.
`-i`	Run the command in interactive mode by invoking the GUI or TUI. See “Installing with the GUI” on page 65.
`-p`	Preview the install task (perform analysis only).
`-r`	Operate on an alternate root directories. See “Installing to an Alternate Root” on page 80.
`-v`	Turn on verbose output to `stdout` and display all activity to the screen.
`-c` *`catalog`*	Store a copy of a response file or other files created by a request script in *catalog*. See “Requesting User Responses (swask)” on page 407.
`-C` *`session_file`*	Save the current option and operand values to *session_file* for re-use in another session. See “Session Files” on page 61.
`-f` *`software_file`*	Read the software selections from *software_file* instead of (or in addition to) software you specify on the command line. See “Software Files” on page 58.
`-Q` *`date`*	Schedules a job for the given date when remote operations are enabled. See “Scheduling Jobs from the Command Line” on page 234 and Chapter 6, “Remote Operations Overview,” on page 189
`-s` *`source`*	Use the software source specified by *source* instead of the default, `/var/spool/sw`. The syntax is: [*`host`*`:`][*`/directory`*]

host may be a host name, domain name, or internet address (for example, `15.1.48.23`). *directory* is an absolute path.

`-S` *`session_file`*
: Use option and operand values saved from a previous installation session and stored in *`session_file`*. See "Session Files" on page 61.

`-t` *`target_file`*
: Read target selections from a *target_file* instead of (or in addition to) targets you specify on the command line. See "Target Files" on page 59.

`-x` *`command_option=value`*
: Sets *`command_option`* to *`value`*, overriding default values or values in options files. See "Changing Command Options" on page 74.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See "Changing Command Options" on page 74.

`software_selections`
: One or more software objects to be installed. See "Software Selections" on page 56.

`target_selections`
: The target on which to install the software selections. See "Target Selections" on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swinstall.

Table 2-3 swinstall Command Options and Default Values

- admin_directory=/var/adm/sw
- agent_auto_exit=true
- agent_timeout_minutes=10000
- allow_downdate=false
- allow_incompatible=false
- allow_multiple_versions=false
- allow_split_patches=false
- ask=false
- autoreboot=false
- autorecover_product=false
- autoremove_job=false
- autoselect_dependencies=true
- autoselect_patches=true
- autoselect_reference_bundles=true
- codeword=
- compress_index=false
- controller_source=
- create_target_path=true
- customer_id=
- defer_configure=false
- distribution_source_directory=/var/spool/sw
- enforce_dependencies=true
- enforce_dsa=true
- enforce_kernbld_failure=true
- enforce_scripts=true
- installed_software_catalog=products
- job_title=
- layout_version=1.0
- log_msgid=0
- logdetail=false
- logfile=/var/adm/sw/swinstall.log
- loglevel=1
- match_target=false
- max_targets=25
- mount_all_filesystems=true
- os_name
- os_release
- patch_filter=software_specification
- patch_match_target=false
- patch_save_files=true
- polling_interval=2
- preview=false
- register_new_root=true
- reinstall=false
- reinstall_files=false
- reinstall_files_use_cksum=true
- retry_rpc=1
- retry_rpc_interval=
- reuse_short_job_numbers=true
- rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- software=
- software_view=all_bundles
- source=
- source_cdrom=/SD_CDROM
- source_tape=/dev/rmt/0m
- source_type=directory
- targets=
- use_alternate_source=false
- verbose=1
- write_remote_files=true

For More Information

See Appendix A, “Command Options,” on page 421 for more information about setting options and a complete listing and description of each option.

Installation Tasks and Examples

This section provides examples of commands for installing software products. Note that The `\*` is an optional shorthand wildcard meaning “all products and filesets or all available software.”

To start an install session via the command line, you must assemble any options (if needed), host and source names, and software selections into a command string. For example:

```
swinstall -p -s softsource -f softlist \
  @ myhost:/mydirectory
```

The `@` *`myhost`*:*`/mydirectory`* is optional if you are installing to your local host and default directory (root).

NOTE

If you do not specify a source, swinstall uses the local host’s default depot directory, `/var/spool/sw`.

- To install a pre-determined list of software products in the file *`mysoft`* that are physically on a CD-ROM (mounted locally at `/mnt/cd`) to the default directory (`/`) on the local host:

  ```
  swinstall -f mysoft -s /mnt/cd
  ```

- To select all software in the default depot (`/var/spool/sw`) located on a host named *`server`* to the default directory on host *`myhost`* and preview the process (`-p`) without actually installing:

  ```
  swinstall -p -s server \* @ myhost
  ```

  A depot location (`:`*`/depot`*) is not specified because it is assumed that the software is located in the default `/var/spool/sw` on *`server`* and will be installed at `/` on *`myhost`*. The `-p` analysis option is explained under “Changing Command Options” on page 87.

- To select all products named `C` and `Pascal` from the default depot on the host named *`sw_server`* and start an interactive GUI session (`-i`):

  ```
  swinstall -i -s sw_server C Pascal
  ```

- To update HP Omniback software (already installed in the default directory on the local host) with a newer version from a CD-ROM mounted at `/mnt/cd`:

  ```
  swinstall -s /mnt/cd Omniback
  ```

Updating to HP-UX 11i

For complete instructions for updating from a previous HP-UX release to HP-UX 11i, use the new update-ux command, as explained in Chapter 2 of *HP-UX 11i Installation and Update Guide*.

This document and complete OS documentation is available on your HP-UX Instant Information CD-ROM and at:

`http://docs.hp.com/hpux/os/11i/`

Installing Patches

swinstall has a variety of patch management features, including a patch management dialog in the GUI. See Chapter 5, “Managing Patches,” on page 163 for complete details on patches and using the swinstall GUI patch features.

Recovering Updated Files

The `autorecover_product` option lets you automatically recover or roll back to original product files if you start an install and the process fails.

When updating files, swinstall removes the original files as they are updated. If an error occurs while swinstall is loading new filesets, the product being loaded is marked as corrupt, the original files are lost, and you must repeat the installation.

By setting the `autorecover_product` option to true, all filesets that are updated are first saved as backup copies. They are not removed until all filesets within the product finish loading. If swinstall terminates because of an error, you can correct the error then re-run swinstall. swinstall automatically continues the load process where it was interrupted.

CAUTION Most HP-UX products have preinstall and postinstall scripts without accompanying undo scripts. This negates any advantage of using the `autorecover_products` option. Use `autorecover_products` only with software that has the associated undo scripts.)

Installing Software That Requires a System Reboot

Software packaged with the `is_reboot` attribute set to true requires the host to be rebooted after the software is installed. However, when installing to alternate root file systems, the host will not be rebooted.

If a local installation entails a reboot, the system reboots the target and the controller, so there is no process left to report success or failure. (SD-UX does not automatically reconnect to the target after a reboot.)

To find out if a software product requires the local host to be rebooted, get a description of the software either from the Software Selection window, using the menu item **Show Description of Software**, or from the Analysis dialog using the **Product Summary** and **Product Description** buttons.

Using Software Codewords and Customer IDs

To protect software from unauthorized installation, HP (and other vendors) use special codewords and customer identification numbers to “lock” the software to a particular owner. These codewords and customer IDs are provided to you when you purchase the software (or receive it as update). HP lists them on the Software Certificate which is packaged with the software.

To properly store the customer_id/codeword for a CD-ROM, you can run swinstall (or swcopy or swlist) on the host serving the CD-ROM. After the codeword has been stored, clients installing software using that host and CD-ROM as a source will no longer require a codeword or customer_id.

SD searches the `.codewords` file on the server that is providing protected software to other hosts. It looks for valid customer_id/codeword pairs. In doing so, SD eliminates the need for you to enter codewords and customer_ids on every host that is “pulling” the software.

This is a time saver if you are updating multiple systems.

SD-UX prompts you for these codewords or numbers prior to the installation of protected software. You can enter or change the numbers via the GUI using the **Add New Codeword** choice from the **Actions** menu in the GUI, or by using the appropriate default option (`-x` *`codeword`*`=`*`xxxx`* and `-x` *`customer_id`*`=`*`xxx`*) on the command line.

For example, if you want to store the codeword *`123456789101bcdf`* (from the *`/CD-ROM`* mount point) and your customer_id was *`xyzCorp`*, you would enter on the command line:

```
swinstall -p -x customer_id=xyzCorp \
  -x codeword=123456789101bcdf \
  -s /CD_ROM
```

(Since the purpose of this command is only to store codewords and customer IDs, the `-p` option runs the command in preview mode so that no actual software installation takes place.)

See Appendix A, “Command Options,” on page 421 for more information on `codeword` and `customer_id` options.

Re-installing Software Distributor

The software product called `SW-DIST` provides all Software Distributor functionality, commands, and tools. If the files that make up SW-DIST are deleted or corrupted, you may need to re-install the product. This process uses the new install-sd command, which is described in Appendix C, “Replacing or Updating SD-UX,” on page 479.

Installing Multiple Versions

Your installation may commonly having multiple versions of a software product installed at various hosts on the network. Multiple installed version let you:

- Back out defective versions (by removing the new version and reconfiguring the old version, if necessary)
- Let users migrate to newer software versions at their own pace

You can decide whether to allow multiple versions by controlling the `allow_multiple_versions` command option. If set to false, installed or configured multiple versions (that is, the same product, but a different revision, installed into a different location) are not allowed. While multiple *installed* versions of software are allowed, multiple *configured* versions are not recommended.

Once multiple versions of software are installed into a location, you can manage them by specifying the product attribute in the software specification of SD-UX commands. (This is as opposed to specifying other version attributes such as revision and architecture). This lets you install old and new versions of software at the same time and configure both versions (if the software packaging supports it).

You can avoid unauthorized, privately installed versions of software by controlling access to the IPD and restricting the use of the swinstall tool.

NOTE Managing multiple versions of a software product on your system requires close attention to the cross-product dependencies that may exist for each version. When you installing multiple versions, make sure you also install multiple versions of the cross-product dependencies. If the dependencies are not relocatable and each version you want to install depends on a different version of the same product, multiple versions of the original product cannot be installed.

Installing to an Alternate Root

Software is usually installed relative to the primary root directory (`/`) but you can also install to an alternate root directory.

The automatic configuration and compatibility filtering that is part of the swinstall command is not performed when installing to an alternate root. You can, however, perform configuration separately from installation by using the swconfig command. See “Configuring Your Installation (swconfig)” on page 82.

Compatibility Filtering and Checking

SD-UX normally filters out software products that are incompatible with any selected targets. Compatible means that the architecture of the hardware matches that required by the software (determined by the system uname attributes). It also means that the OS version is the proper one for the software. The actual check for incompatible software is performed during the selection phase. Compatibility filtering and checking are controlled by the `allow_incompatible` option and depend on the host’s `uname` attributes.

NOTE HP strongly advises that you do not install software that is incompatible unless you are advised to do so by your HP Support representative.

Table 2-4 Product Compatibility

Product attribute	Product value (Pattern to match)	Target Root attribute	
`machine_type`	ia64*	IA	`uname -m`
`machine_type`	9000/*	IA or PA	`uname -m`
`os_name`	HP-UX	HP-UX	`uname -s`
`os_release`	?.11.*	B.11.11	`uname -r`
`os_version`	*	C	`uname -v`

If `allow_incompatible=false` (the default), swinstall restricts the installation of incompatible software and automatically filters the products on the source. The Software Selection window shows only those products compatible with the hardware and OS of all target systems.

If `allow_incompatible=true`, swinstall allows the installation of any software. The GUI displays all products on the source for selection.

You can also use the `-x os_name` and `-x os_release` options to check compatibility. During an OS update, for example, if a system has been installed as 11.0/32 bit and you wish to update to the 64-bit version of HP-UX, you can make the system appear as a 64-bit system for the purpose of compatibility checking against the merged depot by specifying the options `-x os_name=HP-UX:64` and `-x os_release=B.11.00`. (You can also specify these options at a fileset level.)

NOTE Compatibility filtering does not apply to alternate root file systems. You must select software that you know to be compatible with the alternate root.

Configuring Your Installation (swconfig)

The swconfig command runs configuration scripts. Although swinstall and swremove automatically run configuration or unconfiguration scripts, swconfig lets you work independently of these commands. This lets you:

- Execute scripts to address problems if a configuration fails, is deferred, or must be changed.
- Explicitly configure, unconfigure or reconfigure any installed software that has associated configuration scripts.
- Configure or unconfigure hosts that share software located on another host.

Features and Limitations

- swconfig can execute these kinds of scripts:

Configure	Configures installed filesets or products. (Executed by swconfig and swinstall.)
Request	Requests an interactive response from the user as part of the configuration process.
Unconfigure	Undoes configurations performed by configure scripts. For example, removing configuration from the host’s `/etc/profile` or `/sbin/rc` files. This moves the software from the configured state back to installed.

- The swconfig command runs only from the command line interface.
- swconfig configures the host on which the software will run.
- Filesets or products can include configure (unconfigure) scripts.
- swinstall and swremove do not automatically not run configuration scripts when you specify an alternate root directory with these commands. You must run swconfig to configure or unconfigure alternate roots.

- Automatic configuration can also be postponed on software installed to the root directory, `/` (for example, when multiple versions are installed), by using the `defer_configure` command option with swinstall or swremove.
- By default, swconfig only supports configuration of compatible software. You can switch this feature on or off with the `allow_incompatible` option.
- If a fileset relies on another software product for proper operation, that software product must be in a configured state and is controlled by the `enforce_dependencies` option.
- swconfig configures only one version of a fileset at a time, controllable through the `allow_multiple_versions` option.
- swconfig moves software between the installed and configured states.
- swconfig uses dependencies to automatically select software on which to operate (in addition to any software you specify directly). See "Software Dependencies" on page 33 for more information.

NOTE When a swinstall session includes a reboot fileset (such as when you update the core HP-UX operating system to a newer release), the configure scripts are automatically run as part of the system start-up process after the system reboots. You do not have to run swconfig to complete the configuration.

The Configuration Process

The configure process has three phases: selection, analysis, and configuration.

Phase I: Selection In this phase, swconfig resolves the software selections.

Phase II: Analysis In this phase, swconfig determines if the software can be configured successfully (includes checks of software existence, prerequisites). If you execute swconfig with the -p (preview) option, the command stops after completing analysis and does not change anything on the host.

Analysis takes place on the local host. The configuration phase will not take place if any errors occur during analysis. Errors in the analysis phase will only exclude those products that had errors in them. If only `warnings` occur, the task continues.

The sequential analysis tasks on the host are:

1. Initiate analysis.
2. Process software selections:

   Get information from the Installed Product Database and check for compatibility.

   The system checks that all software is compatible with the host's `uname` attributes. This check is controlled by the `allow_incompatible` command option. If it is set to false, the system produces an error; if set to true, it produces a warning.
3. Check state of versions currently installed:
   - If the product is non-existent or corrupt, the task issues an error that says the product cannot be configured and to use swinstall to install and configure this product.
   - If the versions currently installed are not configured and if the `-u` (unconfigure) option is set, the system issues a note that the selected file or fileset is already unconfigured.
   - If the state of versions currently installed is configured, the check is affected by the `reconfigure` option. A note saying the fileset is already configured and will (`reconfigure` is true) or will not (`reconfigure` is false) be reconfigured is issued.
4. Check for configuring a second version:

   If the `allow_multiple_versions` option is set to false, an error is generated stating that another version of this product is already configured and the fileset will not be configured. If the option is set to true, the second version is also configured.
5. Check states of dependencies needed:
   - An error or warning is issued if a dependency cannot be met. This is controlled by the `enforce_dependencies` option. If `enforce_dependencies` is set to true the fileset will not be configured. If `enforce_dependencies` is false, the fileset will be configured anyway.

- If the dependency is a prerequisite, the configuration fails.
- If the dependency is a corequisite, the configuration of this fileset will likely succeed, but the product may not be usable until its corequisite dependency is installed and configured.

Phase III: Configuration

In this phase, the actual software configuration takes place. Configure or unconfigure scripts are executed and the software state is changed from installed to configured (or unconfigured).

The purpose of configuration is to configure the host for the software and configure the product for host specific information. For example, software may need to change the host's `.rc` setup, or the default environment set in `/etc/profile`. Or you may need to ensure that proper codewords are in place for that host or do some compilations. Unconfiguration reverses these steps.

The sequence of configuration tasks is shown below. Products are ordered by prerequisite dependencies, if any. Fileset operations are also ordered by any prerequisites.

1. (Un)configure each product.
2. Run scripts for associated filesets, checking return values.

   If an error occurs, the fileset is left in the installed state. If a warning occurs, the fileset will still be configured.
3. Update the IPD to show the proper installed or configured state.

Configure scripts must also adhere to specific guidelines. For example, these scripts are only executed in the context of the host that the software will be running on, so they are not as restrictive as customized scripts. For more information on scripts, see Chapter 11, "Using Control Scripts," on page 369.

Using swconfig

Syntax

swconfig [-p] [-u] [-v] [-c *catalog*] [-C *session_file*]
[-f *software_file*] [-Q *date*] [-S *session_file*] [-t *target_file*]
[-x *option=value*] [-X *option_file*]
[*software_selections*] [@ *target_selections*]

Options and Operands

-p	Preview a configuration task by running it through the Analysis Phase and then exiting.
-u	Unconfigure the software instead of configuring it.
-v	Turn on verbose output to stdout and display all activity to the screen.
-c *catalog*	Store copy of a response file or files created by a request script. See Chapter 11, “Using Control Scripts,” on page 369.
-C *session_file*	Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.
-f *software_file*	Read a list of software selections from a separate file instead of (or in addition to) the command line. See “Software Files” on page 58.
-Q *date*	Schedules a job for the given date when remote operations are enabled. See “Scheduling Jobs from the Command Line” on page 234 and Chapter 6, “Remote Operations Overview,” on page 189
-S *session_file*	Run the command based on values saved from a previous installation session, as defined in *session_file*. See “Session Files” on page 61.
-t *target_file*	Read a list of target selections from a separate file instead of (or in addition to) the command line. See “Target Files” on page 59.

`-x` *`option=value`*

Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Changing Command Options” on page 87.

`-X` *`option_file`*

Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 87.

`software_selections`

The software objects to be configured. See “Software Selections” on page 56.

`target_selections`

The target of the command. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the options and default values that apply to swconfig.

Table 2-5 swconfig Command Options and Default Values

• admin_directory=/var/adm/sw • agent_auto_exit=true • agent_timeout_minutes=10000 • allow_incompatible=false • allow_multiple_versions=false • ask=false • autoremove_job=false • autoselect_dependencies=true • autoselect_dependents=true • compress_index=false • controller_source= • enforce_dependencies=true • enforce_scripts=true • installed_software_catalog=produ cts • job_title= • log_msgid=0 • logdetail=false	• logfile=/var/adm/sw/swconfig.lo g • loglevel=1 • mount_all_filesystems=true • preview=false • reconfigure=false • reuse_short_job_numbers=true • rpc_binding_info= • ncacn_ip_tcp:[2121] • ncadg_ip_udp:[2121] • rpc_timeout=5 • run_as_superuser=true • select_local=true • software= • targets= • verbose=1 • write_remote_files=false

For More Information

See Appendix A, “Command Options,” on page 421 for more information about setting options and a complete listing and description of each option.

Configuration Tasks and Examples

To configure `productA`, located in the root on the local host:

```
swconfig productA
```

To unconfigure the software selections in the file `mysoft` that are installed in the default directory on the local host:

```
swconfig -u -f mysoft
```

To reconfigure the Omniback product using the default option values:

```
swconfig -x reconfigure=true Omniback
```

To configure a particular version of Omniback:

```
swconfig Omniback,r=2.0
```

To configure the C and Pascal products on the local host:

```
swconfig cc pascal
```

To configure `Product1`, use any associated response files generated by a request script, and save response files under `/tmp/resp1`:

```
swconfig -x ask=true -c /tmp/resp1 Product1
```

To reconfigure the HP Omniback product:

```
swconfig -x reconfigure=true Omniback
```

To configure the version of HP Omniback that was installed at `/opt/Omniback_v2.0`:

```
swconfig Omniback,l=/opt/Omniback_v2.0
```

To unconfigure the software_selections listed in the file `/tmp/install.products` on the hosts listed in the file `/tmp/install.hosts`:

```
swconfig -u -f /tmp/install.products \
         -t /tmp/install.hosts
```

Verifying Your Installation (swverify)

The swverify command verifies depot, installed, or configured software products on the specified host.

Features and Limitations

- Determines whether installed or configured software is compatible with the host on which that software is installed.
- Makes sure that all dependencies (prerequisites, corequisites) are being met (for installed software) or can be met (for copied software).
- Executes verification scripts that check the correctness of the product’s configuration (that is, scripts verify that the installed state of the software is configured).
- Executes fix scripts to correct or report problems with installed software:

Fix	Corrects and reports on problems in installed software. Typical uses are to create missing directories, correct file modifications (mode, owner, group, major, minor), and to recreate missing symbolic links.
Verify	Verifies the configuration of filesets or products, in addition to the standard swverify checks.

  (See Chapter 11, “Using Control Scripts,” on page 369 for more information.)
- Reports missing files, checks all file attributes including permissions, file types, size, checksum, mtime, link source and major/minor attributes.
- Uses dependencies to automatically select software on which to operate (in addition to any software you specify directly). See “Software Dependencies” on page 33 for more information.

The Verification Process

The software verification process has only two phases: selection and analysis.

Phase I: Selection This phase consist of swverify resolving all information on the command line, including all necessary host, software, dependency, and product information.

Phase II: Analysis The analysis phase for swverify takes place on the host. The host’s environment is not modified.

The sequential analysis tasks on each host are:

1. Initiate analysis
2. Process software selections. The system accesses the Installed Products Database (IPD) or depot catalog to get the product information for the selected software:

   For installed software, the system checks that all products are compatible with its `uname` attributes. This check is controlled by the default option `allow_incompatible`:

   - If `allow_incompatible` is set to false, the system produces an error stating that the product is not compatible with the host.
   - If `allow_incompatible` is set to true, a `warning` is issued stating that the product is not compatible.
3. Check for correct states in the filesets (installed, configured or available). For installed software, swverify also checks for multiple versions that are controlled by the `allow_multiple_versions` option:

   - If `allow_multiple_versions` is false, an error is produced that multiple versions of the product exist and the option is disabled.
   - If `allow_multiple_versions` is true, a `warning` is issued saying that multiple versions exist.
4. Check dependencies. An error or warning is issued if a dependency cannot be met. Dependencies are controlled by the `enforce_dependencies` option:

   - If `enforce_dependencies` is true, an error is generated telling you the type of dependency and what state the product is in.

- If `enforce_dependencies` is false, a `warning` is issued with the same information.
- If the dependency is a corequisite, it must be present before the software will operate.
- If the dependency is a prerequisite, it must be present before the software can be installed or configured.

5. Execute verify or fix scripts on installed software in prerequisite order.

   A verify script is used to ensure that the configuration of the software is correct. Possible vendor-specific tasks for a verify script include:

   - Determine active or inactive state of the product.
   - Check for corruption of product configuration files.
   - Check for (in)correct configuration of the product into the OS platform, services or configuration files.
   - Check licensing factors.

   Vendor-supplied scripts are executed and the return values generate an `error` (if 1) or a `warning` (if 2).

   Scripts are executed in prerequisite order.

6. Perform file-level checks for:

   - Contents (mtime, size and checksum) for `control_files`
   - Contents (mtime, size and checksum) for files
   - Missing `control_files`, files and directories
   - Permissions (owner, group, mode) for installed files
   - Proper symlink values

Using swverify

Syntax

`swverify [-d|-r] [-F][-v] [-C` *`session_file`*`] [-f` *`software_file`*`] [-Q` *`date`*`] [-S` *`session_file`*`] [-t` *`target_file`*`] [-x` *`option=value`*`] [-X` *`option_file`*`] [`*`software_selections`*`][@` *`target_selections`*`]`

Options & Operands

`-d`	Operate on a depot rather than installed software. See "Verifying a Depot (swverify -d)" on page 161
`-r`	Operate on an alternate root rather than `/`. Verify scripts are not run.
`-v`	Turn on verbose output to `stdout` and display all activity to the screen. Lets you see the results of the command as it executes.
`-C` *`session_file`*	Run the command and save the current option and operand values to `session_file` for re-use in another session. See "Session Files" on page 61.
`-F`	Run a fix script. See "Fix Scripts" on page 392.
`-f` *`software_file`*	Read a list of software selections from a separate file instead of (or in addition to) the command line. See "Software Files" on page 58.
`-Q` *`date`*	Schedules a job for the given date when remote operations are enabled. See "Scheduling Jobs from the Command Line" on page 234 and Chapter 6, "Remote Operations Overview," on page 189
`-S` *`session_file`*	Run the command based on values saved from a previous verify session, as defined in *`session_file`*. See "Session Files" on page 61.
`-t` *`target_file`*	Read a list of target selections from a separate file instead of (or in addition to) the command line. See "Target Files" on page 59.
`-x` *`option=value`*	Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See "Changing Command Options" on page 93.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 93.

`software_selections`
: The software objects to be verified. See “Software Selections” on page 56.

`target_selections`
: The target of the command. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swverify.

Table 2-6 swverify Command Options and Default Values

- admin_directory=/var/spool/sw
- agent_auto_exit=true
- agent_timeout_minutes=10000
- allow_incompatible=false
- allow_multiple_versions=false
- autoremove_job=false
- autoselect_dependencies=true
- check_contents=true
- check_contents_uncompressed=false
- check_contents_use_cksum=true
- check_permissions=true
- check_requisites=true
- check_scripts=true
- check_volatile=false
- controller_source=
- distribution_target_directory=/var/spool/sw
- enforce_dependencies=true
- installed_software_catalog=products
- job_title=
- log_msgid=0
- logdetail=false
- logfile=/var/adm/sw/swverify.log
- loglevel=1
- mount_all_filesystems=true
- reuse_short_job_numbers=true
- rpc_binding_info=
- ncacn_ip_tcp:[2121]
- ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- software=
- verbose=1

For More Information

See Appendix A, “Command Options,” on page 421 for more information about setting options and a complete listing and description of each option.

Verification Tasks and Examples

To verify an installed fileset *mysoft.myfileset* located on the default depot at *myhosts*, type:

```
swverify -d mysoft.myfileset @ myhosts
```

(The @ sign and the *myhost* target designation are optional because the software being verified located in the default depot on the local host.)

To verify the C and Pascal products that are installed on the local host:

```
swverify C Pascal
```

To verify the HP Omniback product that is installed on the local host and display detailed messages from the process (`-v`) on `stdout`:

```
swverify -v Omniback
```

To verify the 2.0 version of Omniback that is installed on the local host at `/opt/Omniback`:

```
swverify Omniback,r=2.0 @ /opt/Omniback
```

Verify a particular version of HP Omniback:

```
swverify Omniback,1=/opt/Omniback_v2.0
```

Verify the entire contents of a local depot:

```
swverify -d \*@/var/spool/sw
```

3 Managing Installed Software

This chapter presents an overview of managing non-depot software after you have installed it. The swlist, swmodify, and swremove commands help you perform these software management tasks:

Table 3-1 Chapter Topics

Topics:
"Listing Your Software (swlist)" on page 96
"Modifying the IPD (swmodify)" on page 115
"Removing Installed Software (swremove)" on page 122

Listing Your Software (swlist)

The swlist command creates customizable listings of the software products installed on your local host or stored in depots for later distribution.

swlist Features and Limitations

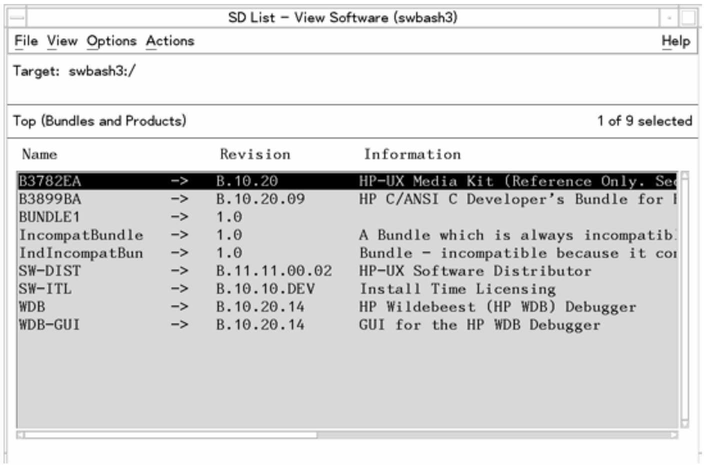
With swlist you can:

- Use an optional GUI.
- Specify the level (bundles, products, subproducts, filesets or files) to show in your list.
- Specify a set of software attributes to display for each level. Software attributes are items of information about products contained in the Installed Products Database or in catalog files. These items can include the product's name or tag, its size (in Kbytes), revision number, etc.
- Create a list of products, subproducts or filesets to use as input to the swinstall or swremove commands.
- Display a table of contents for a software source.
- Display selected software attributes for each level.
- Show the product structure of software selections.
- List software stored in an alternate root directory.
- Display the depots on a specified host.
- List the categories of available or applied patches.
- List the values of a fileset's applied patches.

Using the swlist GUI

The **`swlist -i`** command starts a swlist GUI program that lets you interactively list software and display software information. The **`swlist -i -d`** command lets you display information about the software available in a depot or on a physical media.

Figure 3-1 **The swlist Browser**



```
Target:  swbash3:/

Top (Bundles and Products)                                1 of 9 selected

Name                 Revision          Information
B3782EA         ->   B.10.20           HP-UX Media Kit (Reference Only. Se
B3899BA         ->   B.10.20.09        HP C/ANSI C Developer's Bundle for
BUNDLE1         ->   1.0
IncompatBundle  ->   1.0               A Bundle which is always incompatib
IndIncompatBun  ->   1.0               Bundle - incompatible because it co
SW-DIST         ->   B.11.11.00.02     HP-UX Software Distributor
SW-ITL          ->   B.10.10.DEV       Install Time Licensing
WDB             ->   B.10.20.14        HP Wildebeest (HP WDB) Debugger
WDB-GUI         ->   B.10.20.14        GUI for the HP WDB Debugger
```

- Bundles and products are the default top-level display.
- To open an item on the list, double-click on the item.
- Double-clicking on a file displays the file attributes.

Searching and Moving Through the List

The following features help you search and move through the list:

- To search the current list, select **File→Search...**
- To display a pop-up menu of viewing options for an item, right-click on the item. The pop-up options are:
  - — **Open Item** to show the contents of the item.
  - — **Close Level** to close the current item and displays the next higher level of objects.
  - — **Show Description of Software...** to display attribute information about the current item.

Changing the View

Use the **View** menu to change the columns displayed, select filters, and sort information:

- **Columns** displays the Columns Editor. You can choose which columns of software information to display (i.e. software name, revision number, information, size in Kbytes, architecture, category, etc.) and their order.
- **Filter...** displays a dialog from which you can filter the display list with logical and relational operators for each field.
- **Sort...** lets you select sort fields, order, and criteria for the information displayed.
- **Change Software View** lets you toggle between a top-level view and a products view.
- **Change Software Filter...** lets select from a list of predefined filters. (Only applies to top-level software objects.)

Performing Actions

Use the **Actions** menu to open and close items on the display, show logfile information, and show software descriptions:

- **Open Item** opens an item. (Same as double-clicking on the item.)
- **Close Level** closes the current level. (Same as double-clicking on **..(go up)**.
- **Change Target** opens a dialog box that lets you enter a path to select an alternate root (for `swlist -i`) or alternate depot (for `swlist -i -d`).
- **Show Logfile** displays the system logfile.
- **Show Audit Log** displays software depot audit information stored in the audit log (for `swlist -i -d` only). See “Source Depot Auditing” on page 160 for more information.
- **Show Description of Software** displays attribute information about the currently selected item.

Using the Command Line

Syntax

`swlist [-d|-r]] [-i] [-R] [-v] [-a` *`attribute`*`] [-c` *`catalog`*`]`
`[-C` *`session_file`*`] [-f` *`software_file`*`] [-l` *`level`*`] [-s` *`source`*`]`
`[-S` *`session_file`*`] [-t` *`target_file`*`] [-x` *`option=value`*`]`
`[-X` *`option_file`*`] [`*`software_selections`*`] [@` *`target_selections`*`]`

Options and Operands

`-d` List products available from a depot. See “Listing the Contents of a Depot (swlist -d)” on page 159.

`-i` Start the GUI. (See “Using the swlist GUI” on page 97.)

`-r` List products on an alternate root (instead of `/`).

`-R` Shorthand for
`-l bundle -l product -l subproduct -l fileset`

`-a` *`attribute`* Displays a specific attribute. To display multiple attributes, specify multiple `-a` options. To list the full set of attributes for a software object, use the -v option. Note that the tag attribute is always displayed for products, subproducts, and filesets. The path (filename) attribute is always displayed for file objects. This option does not apply if you use the `-c` option.

`-v` List all attributes for an object if no `-a` option is specified. (Vendor-defined attributes are not included.) The output lists one attribute per line in the format:

```
attribute_name      attribute_value
```

`-c` *`catalog`* Writes full catalog structure information into the directory specified by *`catalog`*. You can use this information for distributions and to list installed software catalog information. All attributes down to the file level and control scripts are written. If you use this option, the `-a` and `-l` options do not apply. See “Requesting User Responses (swask)” on page 407.

`-C` *`session_file`*
Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.

`-f` *`software_file`*
: Read a list of software selections from a separate file instead of (or in addition to) the command line. See "Software Files" on page 58.

`-l` *`level`*
: List all software objects *down to* the specified level: `depot`, `bundle`, `product`, `subproduct`, `fileset` or `file`. (See the section "Listing Software by Levels" on page 107 for more information on levels.) You can use only one level designation per command. You cannot use software names, subproduct names, etc. to specify levels. This option does not apply if you use the `-c` option.

Table 3-2 The `-l` Options

Option	Action
`swlist -l root`	shows the root level (roots on the specified target hosts)
`swlist -l shroot`	Shows the shared roots
`swlist -l prroot`	Shows the private roots
`swlist -l bundle`	Shows only bundles
`swlist -l product`	Shows only products
`swlist -l subproduct`	Shows products and subproducts
`swlist -l fileset`	Shows products, subproducts and filesets
`swlist -l file`	Shows products, subproducts, filesets, files and numbers (used in software licensing).
`swlist -l category`	Shows all categories of available patches for patches that have included category objects in their definition.
`swlist -l patch`	Shows all applied patches.

`-s` *`source`*
: Specify which software source is to be listed. The default source type is a directory or depot (usually `/var/spool/sw`) on the local host. The syntax is:

  [*`host`*][`:`][*`/directory`*]

  A host may be specified by its host name, domain name, or internet address. A directory must be specified by an absolute path.

`-S` *`session_file`*
: Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-t` *`target_file`*
: Read a list of target selections from a separate file instead of (or in addition to) the command line.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Changing Command Options” on page 102.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 102.

`software_selections`
: The software objects to be listed. See “Software Selections” on page 56.

`target_selections`
: The target of the command. (For swlist, *target_selections* are just another way to list software selections.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swinstall.

Table 3-3 swlist Command Options and Default Values

- admin_directory=/var/adm/sw
- agent_timeout_minutes=10000
- codeword=
- customer_id=
- distribution_target_directory= /var/spool/sw
- installed_software_catalog= products
- layout_version=1.0
- level=
- log_msgid=1
- one_liner=revision title
- patch_one_liner=title patch_state
- rpc_binding_info= ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- show_superseded_patches=fals e
- software=
- software_view=all_bundles
- targets=
- verbose=1

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

Software Listing Tasks and Examples

To run the `swlist` interactive interface:

`swlist -i @ host1`

To use interactive `swlist` to view a depot:

`swlist -i -d @ /tmp/depot`

To produce a list of the software (by name) installed at root (`/`) on your local host, you would simply type:

`swlist`

Which might produce a listing on your display like this:

```
# Initializing...
# Contacting target "xxyyzz"...
#
# Target:  xxyyzz:/

# Bundle(s):

B3782CA B.11.00 HP-UX Media Kit (Reference Only. See Descr.)
B3898AA B.11.00 HP C/ANSI C Developer’s Bundle for HP-UX 11.00
HPUXEngRT  B.11.00  English HP-UX Run-time Environment

# Product(s) not contained in a Bundle:

HMS               1.01
OBAM5_0            B.11.00        ObAM 5.0
```

Using `swlist` with no options set and no software selected gives you a listing of all software bundles plus all products that are not part of a bundle. Adding the `-d` option gives you the same listing of software residing in the default depot on your local host.

In the following examples, `swlist` requests are sent to the standard output. All examples assume the `one_liner=` default is “revision size title” and the `level=` default is “product.”

- To list the contents of the local tape depot, /dev/rmt/0m, type:

  `swlist -d @ /dev/rmt/0m`

  — *or* —

  `swlist -s /dev/rmt/0m`

This produces the following output

```
AUDIT      3.5   9834  Trusted Systems Auditing Utils
COMMANDS   1.7   4509  Core Command Set
C-LANG     2.5   5678  C Programming Language
NETWORKING 2.1   9072  Network Software
KERNEL     1.4  56908  Kernel Libraries and Headers
VUE        1.3   5489  Vue (Instant Ignition Release)
WINDOWS   2.06 10423  Windowing Products
```

- List all the media attributes of the local tape depot, /dev/rmt/0m:

  `swlist -v -l depot @ /dev/rmt/0m`

  — *or* —

  `swlist -vl depot -s dev/rmt/0m`

```
type        distribution
tag         CORE OS
description HP-UX Core Operating System Software Disk
number      B2358-13601
mod_date   June 1998
```

- List the README file for product, OS_CORE installed on the local host:

  `swlist -a readme OS-Core | more`

```
readme:
 ****************
 * Introduction *
 ****************

The Release Notes for HP-UX Release X.0 contain an
overview of the new/changed product features that
are included in the release.  For detailed
information about these features, refer to the
appropriate product manuals.  This document does not
contain information about software changes made as a
result of a Service Request; that information may be
found in the Software Release Bulletin (SRB) for Release
 X.0.
 ********************
 * Hardware Support *
 ********************
 The HP 9000 Model XXX is no longer supported.
...
```

- List the products stored in the software depot on host1 located at `/swmedia`. For this example assume the `swlist one_liner` is: “title size architecture”:

  `swlist -d @ host1:/swmedia`

  ```
  FRAME     Frame Doc. Pkg 2319  HP-UX_9000_Series_AorB
  FRAME     Frame Doc. Pkg 2458  OSF1_9000_Series_1.0
  ME30      3-D Mech. Eng  5698  HP-UX_9000_Series300_AorB
  SOFTBENCH Dev Env        4578  HP-UX_9000_Series300
  TEAMWORK  Design/Analysis 3478 HP-UX_9000_Series 300/400
  ```

  (Note that the media contains two revisions of the FRAME product.)

Using Options to Change List Appearance

You can control the appearance and content of your lists by changing list default values in the options files. Instead of repeatedly specifying the software levels and attributes each time you invoke swlist, you can use:

`level` This option pre-determines what level to list: product, subproduct, fileset or file. For example, by setting this default to `level=fileset`, future swlist commands would always list everything down to and including filesets for each host, depot or product selected.

`one_liner="attribute attribute attribute"`
This option specifies the attributes (revision, size, title, etc.) displayed as the default listing. These attributes are separated by <tab> or <space> and enclosed in quotes (" "). You can choose multiple attributes but not all attribute may exist for all applicable software levels (product, subproduct, or fileset). For example, the software attribute *`title`* is available for bundles, products, subproducts and filesets, but the attribute *`architecture`* is only available for products.

In the absence of the `-v` or `-a` option in your command, swlist displays the information as described in the `one_liner` default for each software object level (bundle, products, subproducts and filesets), not for files.

Listing Attributes

You may specify only one attribute per `-a` option. However, the `tag` attribute is always included by default, so specifying `-a` *`revision`* lists all product names *and* their revision numbers.

For example, to list whether software bundles on a CD-ROM (mounted to the directory `/SD_CDROM`) require a codeword or not, use the command:
`swlist -d -a is_protected @ /SD_CDROM`

An attribute containing a large amount of information (for example, a README) is physically stored as a separate file and is displayed by itself if `-a` *`README`* is requested.

Refer to the *sd*(4) manpage for a full list of SD-UX attributes.

Creating Custom Lists

The swlist options and defaults allow you to create lists to fit your specific requirements. These lists can be as simple as listing the software products installed on your local host or as complex as a multiple column listing of files, filesets, subproducts, products and bundles installed.

For example, if you were to change the `one-liner` option on the command line, the command:

```
swlist -x one_liner="name revision size title"
```

produces this list of all the products installed on the local host:

```
RX         1.98    9845    RX X Terminal - all software
ALLBASE       8.00.1  6745     Database Products
C-LANG        2.5     5678     Programming Language
DIAGNOSTICS   2.00    56870    Hardware Diagnostic Programs
DTP68         2.00    26775    Desktop Publishing
LISP-LANG     8.00.1  90786    LISP Programming Language
WINDOWS      2.06    10423   Windowing Products
```

This listing shows, in columns from left to right, the product's tag, its revision number, its size in Kbytes and its title or full name.

NOTE Whatever you specify in the command line for software level and attributes will override the values in the default option files.

You can also change the `one_liner` default value to {revision size title} in the defaults file. Then a listing of the C-LANG products on host2 would be as follows:

```
swlist C-LANG @ host2
```

```
C-LANG.C-COMPILE 8.0    1346       C Compiler Components
C-LANG.C-LIBS    8.0    2356       Runtime Libraries
C-LANG.C-MAN     8.0    1976       Programming Reference
```

Listing Patches

You can use swlist to list software patches and their status. See “Listing Patches” on page 178 for more information.

Using Software Codewords and Customer IDs

The swlist command may prompt you for codewords if you try to view codeword protected software. You can also enter new codewords from the command line or from the GUI. This process is identical to that used by swinstall. See “Using Software Codewords and Customer IDs” on page 78 for more information.

Listing Software by Levels

The `-l` *`level`* option lets you list all software objects *down to* the specified level: `depot`, `bundle`, `product`, `subproduct`, `fileset` or `file`.

Choose a level as a starting point and list items only down to that level.

Table 3-4 **The `-l` Options**

Option	Action
`swlist -l root`	Shows the root level (roots on the specified target hosts)
`swlist -l shroot`	Shows the shared roots
`swlist -l prroot`	Shows the private roots
`swlist -l bundle`	Shows only bundles
`swlist -l product`	Shows only products

Table 3-4 The `-l` Options (Continued)

Option	Action
`swlist -l subproduct`	Shows products and subproducts
`swlist -l fileset`	Shows products, subproducts and filesets
`swlist -l file`	Shows products, subproducts, filesets, files and numbers (used in software licensing).
`swlist -l category`	Shows all categories of available patches for patches that have included category objects in their definition.
`swlist -l patch`	Shows all applied patches.

The starting point for a software list is always taken from the operands in the `-l` and `-a` options (or from the `level` or `one_liner` options). You must decide what levels you want and what software attributes to list in addition to the product name.

NOTE Examples in the following sections do not include a value for the `one_liner` option.

Specifying Product Level Specifying a level for a given software selection causes swlist to list the objects at that level plus all those that are *above* that level. Upper levels will be commented with a # sign. Therefore, only the level specified (product, subproduct, fileset or file) will be uncommented. This allows the output from swlist to be used as input to other commands. The exceptions are:

1) a list that contains only files; file-level output is not accepted by other commands

2) a list that contains software attributes (`-a` and `-v`).

For example, if you wanted to see all the *`products`* installed on your local host, your command would be:

```
swlist -l product
```

and the listing would look like this:

```
NETWORKING
SAM
OPENVIEW
PRODUCT A
SOFTWARE Z
PRODUCT B
.
.
.
```

Note that the product names are uncommented because that was the level you requested to display and there are no levels above.

Specifying Subproduct Level For this example, on the local host, the NETWORKING product contains the subproducts ARPA and NFS and you want to see how big each object is (in Kbytes).

```
swlist -l subproduct -a size NETWORKING
```

```
# NETWORKING              9072
  NETWORKING.ARPA         4412
  NETWORKING.NFS          4660
```

The list does not show the files or filesets because you didn’t specify that level on the command line.

If you wanted to see the names and revision numbers for the NETWORKING product on the local host, the command would be:

```
swlist -l subproduct -a revision NETWORKING
```

Remember, the product name is always assumed; you don’t have to specify it in the `-a` option.

Specifying Fileset Level An example of using the `-l` option to generate a listing that includes all filesets for the product NETWORKING on the local host and a descriptive title for each:

`swlist -l fileset -a title NETWORKING`

```
# NETWORKING                  Network Software
  NETWORKING.ARPA-INC         ARPA include files
  NETWORKING.ARPA-RUN         ARPA run-time commands
  NETWORKING.ARPA-MAN         ARPA manual pages
  NETWORKING.LANLINK          CORE ARPA software
  NETWORKING.NFS-INC          NFS include files
  NETWORKING.NFS-RUN          NFS run-time commands
  NETWORKING.NFS-MAN        NFS manual pages
```

Again, note the commented lines (#) representing the subproduct (`NETWORKING.ARPA` and `NETWORKING.NFS`) and product (`NETWORKING`) levels. The other lines are filesets.

Specifying Files Level An example of the `-l` option to generate a comprehensive listing that includes all files for the subproduct NETWORKING.ARPA:

`swlist -l file NETWORKING.ARPA`

```
# NETWORKING.ARPA
# NETWORKING.ARPA_INC
  NETWORKING.ARPA_INC:/usr/include/arpa/ftp.h
  NETWORKING.ARPA_INC:/usr/include/arpa/telnet.h
  NETWORKING.ARPA_INC:/usr/include/arpa/tftp.h
  NETWORKING.ARPA_INC:/usr/include/protocols/rwhod.h
.
.
.
# NETWORKING.ARPA_RUN
  NETWORKING.ARPA_RUN:/etc/freeze
  NETWORKING.ARPA_RUN:/etc/ftpd
  NETWORKING.ARPA_RUN:/etc/gated
  NETWORKING.ARPA_RUN:/etc/named
.
.
.
# NETWORKING.ARPA_MAN
  NETWORKING.ARPA_MAN:/usr/man/man8/ftpd
  NETWORKING.ARPA_MAN:/usr/man/man8/gated
```

Note that the commented lines represent the requested level (`NETWORKING.ARPA`) plus one level up (fileset) from the specified file level (`NETWORKING.ARPA_INC`, `NETWORKING.ARPA_RUN` and `NETWORKING.ARPA_RUN` are all filesets). The uncommented lines are files.

Depot Lists Another class of objects that swlist can display are depot lists. This allows you to list all the registered depots residing on a host. To do this, you can use a combination of the `-l` *`depot`* option:

Table 3-5 **Listing Depots**

`swlist` syntax	**result**
`swlist -l depot`	list all depots on the local host
`swlist -l depot @ hostA`	list all depots on hostA
`swlist -l depot -v @ hostB`	list, in verbose mode, all depots on hostB

Verbose List The `-v` option causes a verbose listing to be generated. A verbose listing is used to display all attributes for products, subproducts, filesets or files.

The verbose output lists each attribute with its name (keyword). The attributes are listed one per line. Given the length of this listing, you could post-process (filter) the output with `grep` and/or `sed` to see specific fields.

Attributes for a particular software level are displayed based on the software product name given with the `swlist` command. For example, **`swlist -v NETWORKING`** gives:

```
tag                 NETWORKING
instance_id         7869
control_directory
size                9072
revision            2.1
title               Network Software
mod_time
directory
```

```
vendor.information  Hewlett-Packard Company
is_locatable        true
architecture        HP-UX_9000
machine_type        9000
os_name             HP-UX
target.os_release   B.11.00*
```

If the `-v` option is used with the `-l` option, the cases are:

- To display all attributes for a bundle, use `swlist -v -l bundle`.
- To display all attributes for a product, use `swlist -v -l product`.
- To display all attributes for products and subproducts, use `swlist -v -l subproduct`.
- To display all attributes for products, subproducts and filesets, use `swlist -v -l fileset`.
- To display all attributes for products, subproducts, filesets and files, use `swlist -v -l file`.

The table below provides a sample listing of the kinds of attributes that swlist will display. Not all these attributes exist for each software level or object. This list may change depending on vendor-supplied information. Do not use this list as the official list of all attributes. To get a complete list of the attributes for a particular level or object, use the format:

```
swlist -v -l level
```

(see example above) or use

`swlist -v` *`software_selections`*

(see example below).

Table 3-6 Sample Attributes

Attribute	**Description**
architecture	Describes the target system(s) supported by the product
category	Type of software
copyright	Copyright information about the object
mod_time	Production time for a distribution media

Table 3-6 Sample Attributes (Continued)

Attribute	Description
description	Detailed descriptive information about the object
instance_id	Uniquely identifies this software product
title	Long/official name for the object
mode	Permission mode of the file
mtime	Last modification time for the file
owner	Owner of file (string)
path	Full pathname for the file
corequisite	A fileset that the current fileset needs (`configured`) to be functional
prerequisite	A fileset that the current fileset needs to install or configure correctly
readme	Traditional readme-like information, release notes, etc.
revision	Revision number for an object
size	Size in bytes; reflects the size of all contained filesets
state	Current state of the fileset

Here are some examples of verbose listings:

This command on the local host:

`swlist -v -l file NETWORKING.ARPA-RUN`

produces this listing:

```
#NETWORKING.ARPA
tag:           ARPA-RUN
instance_id    1
revision       1.2
title          ARPA run_time commands
size           556
state          configured
```

```
corequisite   NETWORKING.LANLINK
is_kernel     true
file          etc/freeze
path          /etc/freeze
type          f
mode          0755
owner         bin
group         bin
uid           2
gid           2
mtime         721589735
size          24
file          etc/ftpd
path          /etc/ftpd
type          file
mode          0555
owner         bin
group         bin
uid           2
gid           2
mtime         721589793
size          9
...
```

This command:

swlist -v NETWORKING.ARPA-RUN

produces the following listing:

```
# NETWORKING.ARPA
fileset
tag           ARPA-RUN
instance_id   1
revision      1.2
title         ARPA run_time commands
size          556
state         configured
corequisite   NETWORKING.LANLINK
is_kernel     true
mod_time     733507112
```

Modifying the IPD (swmodify)

SD-UX keeps track of software installations, products, and filesets on your system with the Installed Products Database (IPD) for installed software and with catalog files for software in depots.

Both the IPD and catalog files are created and constantly modified by other SD-UX operations (swinstall, swcopy, and swremove), they are not directly accessible if you want to change the information they contain. If you need to edit the information in either the IPD or in any depots' catalog files, you must use the swmodify command.

The swmodify command adds, modifies, or deletes software objects or attributes defined in a software depot, primary root or alternate root. It is a direct interface to a depot's catalog files or a root's Installed Products Database. It does not change the files that make up the object, it only manipulates the information that *describes* the object.

Using swmodify, you can

- Add new bundle, product, subproduct, fileset, control script or file definitions to existing objects
- Remove the description of software objects from a depot catalog file or root IPD
- Change attribute values for any existing object.
- Define attributes for new objects that you add.

The equivalent IPD files for a depot are called catalog files. When a depot is created or modified using swcopy, catalog files are built (by default in `/var/spool/sw/catalog`) that describe the depot and its contents.

IPD Contents

Located in the directory `/var/adm/sw/products`, the IPD is a series of files and subdirectories that contain information about all the products that are installed under the root directory (`/`). This information includes "tags" or product names, one-line title fields, paragraph-or-longer description text, long README files, copyright information, vendor information and part numbers on each product installed. In addition, the IPD contains revision information and a user-targeted architecture field including the four uname attributes (operating system name, release, version and hardware machine type). Here is what the IPD `INFO` file for a product called “Accounting” looks like:

```
fileset
tag ACCOUNTNG
data_model_revision 2.4
instance_id 1
control_directory ACCOUNTNG
size 292271
revision B.11.00
description Vendor Name: Hewlett-Packard Company
Product Name: Accounting
Fileset Name: ACCOUNTING

Text: "HP-UX System Accounting feature set. Use these
features to
gather billing data for such items as disk space
usage, connect time or CPU resource usage.
"
timestamp 797724879
install_date 199504121614.39
install_source hpfclc.fc.hp.com:/release/11.00_gsL/goodsyste
m state configured
ancestor HPUX10.20.ACCOUNTNG
corequisite OS-Core.CMDS-MIN,r>=B.11.00,a=HP-UX_B.11.00_32/6
4,fa=HP-UX_B.11.00_32/64,v=HP
```

Catalog files are the equivalent IPD files but they are for software stored in a depot. When a depot is created or modified using swcopy, these files are created and placed in the specified depot (or in the default `/var/spool/sw` depot). They describe the depot and its contents.

The swinstall, swconfig, swcopy, and swremove tasks automatically add to, change and delete IPD and catalog file information as the commands are executed. swlist and swverify tasks read the IPD information and use it to affect command behavior.

The IPD also contains the `swlock` file, which manages simultaneous read and/or write access to software objects.

Using swmodify

Syntax

swmodify [-d] [-p] [-r] [-u] [-v [-V] [-a *attribute=[value]*]
[-c *catalog*][-C *session file*] [-f *software_file*]
[-P *pathname_file*] [-s *product_specification_file*]
[-S *session_file*] [-x *option=value*][-X *option_file*]
[*software_selections*] [@ *target_selection*]

Options and Operands

-d	Perform modifications on a depot (not on a primary or alternate root). Your *target_selection* must be a depot.
-p	Previews a modify session without changing anything within the *target_selection*.
-r	Perform modifications on an alternate root instead of the primary root. Your *target_selection* must be an alternate root.
-u	If no -a *attribute* options are specified, then delete the specified *software_selections* from within your *target_selection*. This action deletes the definitions of the software objects from the depot catalog or Installed Products Database. If -a *attribute* options are specified, then delete them from within the given *target_selection*.
-v	Turns on verbose output to stdout. (The swmodify logfile is not affected by this option.)
-V	Lists all the SD *layout_versions* this command supports.
-a *attribute=value*	Add, change, or deletes the *attribute value*. Otherwise, it adds/changes the attribute for each *software_selection* by setting it to the given *value*. Multiple -a options can be specified. Each attribute modification will be applied to every *software_selection*. The -s and -a options are mutually exclusive: the -s option cannot be specified when the -a option is specified.

You cannot use the `-a` option to change the following attributes: `tag`, `revision`, `instance_id`, `vendor_tag`, `corequisite` or `prerequisite`.

`-c` *`catalog`* Writes full catalog structure information into the directory specified by *`catalog`*. All attributes down to the file level and control scripts are written. See "Requesting User Responses (swask)" on page 407.

`-C` *`session_file`*

Run the command and save the current option and operand values to a session_file for re-use in another session. See "Session Files" on page 61.

`-f` *`software_file`*

Read a list of software selections from a separate file instead of (or in addition to) the command line. See "Software Files" on page 58.

`-P` *`pathname_file`* Specifies a file containing the pathnames of files being added to or deleted from the IPD.

`-s` *`product_specification_file`*

The source Product Specification File (PSF) describes the product, subproduct, fileset, and/or file definitions that will be added or modified by swmodify.

If you specify a *`product_specification_file`*, swmodify selects the individual *`software_selections`* from the full set that is defined in the PSF. If no *`software_selections`* are specified, then swmodify will select *all* of the software defined in the PSF. The software selected from a PSF is then applied to the *`target_selection`*, with the selected software objects either added to, modified in, or deleted from it.

If a PSF is not specified, then *`software_selections`* must be specified. swmodify will select the *`software_selections`* from the software defined in the given (or default) *`target_selection`*.

The product specification file (PSF) for swmodify uses the same swpackage PSF format as defined in "Creating a Product Specification File (PSF)" on page 307.

`-S` *`session_file`*
: Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Changing Command Options” on page 120.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 120.

`software_selections`
: The software objects for which information will be modified. See “Software Selections” on page 56.

`target_selection`
: A single, local *`target_selection`*. (See “Target Selections” on page 58.) If you are operating on the primary root, you do not need to specify a *`target_selection`* because the target `/` is assumed.

: When operating on a software depot, the *`target_selection`* specifies the path to that depot. If the `-d` option is specified and no *`target_selection`* is specified, then the default *`depot_directory`* is assumed.

NOTE In general, use caution when using the `-u` option with the `-a` option. If `-u` is used and `-a` is also specified, the `-a` option deletes the *`attribute`* from the given *`software_selections`* (or deletes the *`value`* from the set of values currently defined for the *`attribute`*).

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swmodify.

Table 3-7 swmodify Command Options and Default Values

• admin_directory=/var/adm/sw	• logfile= /var/adm/sw/swmodify.log
• compress_index=false	• loglevel=1
• control_files=	• patch_commit=false
• distribution_target_directory=/var/spool/sw	• run_as_superuser=true
• files=	• software=
• installed_software_catalog=products	• source_file=
• layout_version=1.0	• targets=
• log_msgid=0	• verbose=1
• logdetail=false	

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

swmodify Tasks and Examples

Here are some examples of how you can use swmodify to change catalog files or IPDs:

Adding Information to the IPD

To add descriptions of files `/tmp/a`, `/tmp/b`, and `/tmp/c` to an existing fileset:

```
swmodify -x files=/tmp/a /tmp/b /tmp/c  PRODUCT.FILESET
```

If a control script adds new files to the installed file system, the script can use swmodify to make a record of the new files.

Changing Existing IPD Information

To create some new bundle definitions for products in an existing depot:

```
swmodify -d -s new_bundle_definitions \
         \* @ /mfg/master_depot
```

If a product provides a more complex configuration process, a script can set the fileset’s state to `configured` upon successful completion.

To change the values of a fileset’s attributes:

```
swmodify -a state=installed PRODUCT.FILESET
```

To change the attributes of a depot:

```
swmodify -a title=Master Depot \
         -a description=/tmp/mfg.description \
         @ /mfg/master_depot
```

Defining New Objects

You can import an existing application (not installed by SD-UX) by constructing a simple Product Specification File (PSF) describing the product and then invoke swmodify to load that definition into the IPD.

To create a new fileset definition (if the PSF contains file definitions, then add those files to the new fileset):

```
swmodify -s new_fileset_definition
```

Removing Installed Software (swremove)

The swremove command removes software that has been installed on a host. Before its removal, the software is first unconfigured. swremove also removes software products that have been copied to a software depot.

swremove Features and Limitations

- Removes files from the specified location. It removes symbolic links, but not the targets of symbolic links. It also lists busy files that were not removed.
- Automatic use of dependencies to automatically select software on which to operate (in addition to any software you specify directly).
- Ability to run control scripts as part of the removal:

Unconfigure	Undoes host configuration performed by configure scripts.
Checkremove	Analyzes each target to determine if removal and unconfiguration can take place. If this check fails, an object cannot be removed.
Preremove	Performs additional file operations, such as removing files created by a preinstall script.
Postremove	Performs additional remove operations (such as restoring "rollback" files) immediately after a fileset or product has been removed.

  For more information, see Chapter 11, “Using Control Scripts,” on page 369.
- swremove does not perform automatic unconfiguration when you remove software from alternate roots.

Using the swremove GUI

This section provides an overview of the swremove GUI.

- In general, all information presented in “Removing Installed Software (swremove)” on page 122 also applies to the swinstall GUI.
- This section refers to additional information about standard GUI elements, discussed in “Using the GUI and TUI Commands” on page 35.
- All information in this section also applies to the TUI program unless otherwise noted. See “The Terminal User Interface” on page 35.

The swremove command behaves slightly differently when removing from primary root file systems, alternate root file systems, and depots. Interface changes for depot remove operations are summarized in “Removing Software from Depots” on page 162.

There are three steps in the copy process:

Table 3-8 GUI Removal Steps

I. Start-Up	Start the swremove GUI.
II. Select Software	Choose the software to remove.
III. Analysis (Preview)	Analyze (preview) the removal to determine if the selected software can be successfully removed.
IV. Removal	Perform the actual removal.

Step I: Start-Up

To start the GUI or TUI for an install session, type:

```
/usr/sbin/swremove
```

The GUI is automatically invoked *unless* you also specify software on the command line. To invoke the GUI *and* specify software, include the `-i` option. For example, to use the GUI for a preview (analysis only) session with `BUNDLE1`, type:

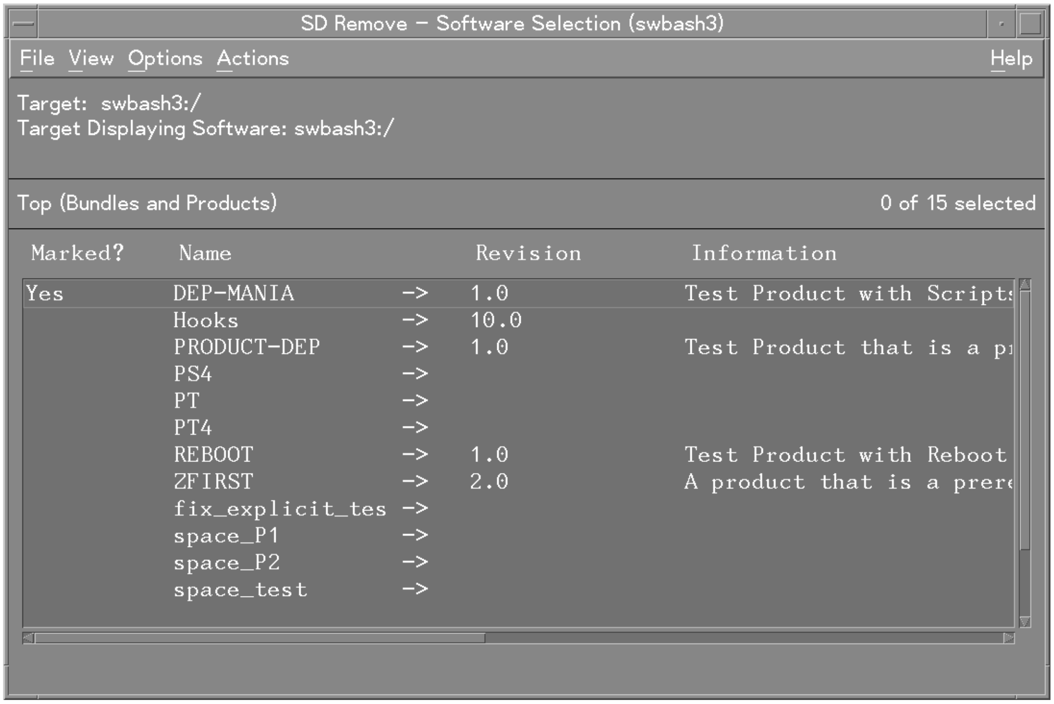
```
swremove -i -p /MyDepot/BUNDLE1
```

The Software Selection window appears.

Step II: Selecting Software

In this step, you use the Software Selection window to select the software you want to install.

Figure 3-2 swremove Software Selection Window


SD Remove – Software Selection (swbash3)
File View Options Actions Help

```
Target:  swbash3:/
Target Displaying Software: swbash3:/

Top (Bundles and Products)                              0 of 15 selected

Marked?     Name                    Revision     Information
Yes         DEP-MANIA          ->   1.0          Test Product with Script
            Hooks              ->   10.0
            PRODUCT-DEP        ->   1.0          Test Product that is a p
            PS4                ->
            PT                 ->
            PT4                ->
            REBOOT             ->   1.0          Test Product with Reboot
            ZFIRST             ->   2.0          A product that is a prer
            fix_explicit_tes   ->
            space_P1           ->
            space_P2           ->
            space_test         ->
```

1. Select software from the object list:

   a. Highlight an item

   b. Select **Actions→Mark For Remove**

   — or —

   Right-click to display the pop-up, then select **Mark For Remove**

   The `Marked?` flag in the object list changes to `Yes` to match your selection. (The flag `Partial` may appear if you select only some component of a software object.)

2. (Optional) Use choices from the **Actions** menu to make additional software selections:

   - **Change Target** lets you select an alternate root from which to remove software.
   - **Add Software Group** lets you recall and re-use a group of previously saved software selections.
   - **Save Software Group** saves the current list of marked software as a group. SD stores the group definition in `$HOME/.sw/software/` or a directory you specify.
   - **Show Description of Software** (available only for a single item highlighted in the object list) displays more information on the selected software.

3. Select **Actions→Install** to start the analysis (preview) step. The Analysis dialog appears.

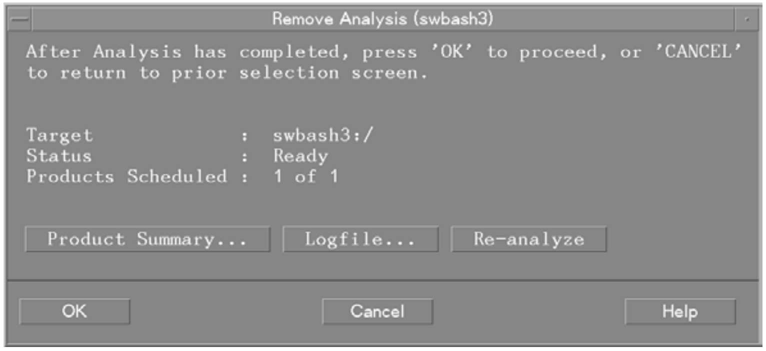
Step III: Analysis (Preview)

In this step, SD-UX analyzes the software you have selected.

The Remove Analysis dialog displays status information about the analysis process. When the analysis is complete and the host status shows `Ready`, click **OK** to start the actual installation (see "Step IV: Removal" on page 127). The Analysis dialog is then replaced by the Remove Window.

(If you started a preview session, the install stops after the analysis. Clicking **OK** returns you to the Software Selection window.)

Figure 3-3 **Remove Analysis dialog**



```
After Analysis has completed, press 'OK' to proceed, or 'CANCEL'
to return to prior selection screen.

Target             :  swbash3:/
Status             :  Ready
Products Scheduled :  1 of 1
```

After analysis, if any of the selected software can be removed, the status indicates `Ready` or `Ready with Warnings`. If none of the selected software can be removed, the status indicates `Excluded from task`.

The `Products Scheduled` column shows the number of products ready for removal out of all products selected. The total products ready includes those products that are:

- Marked because of dependencies
- Marked inside of bundles
- Partially and wholly marked

A product may be automatically excluded from the removal if an error occurs with that product. Removal cannot proceed if the host target is excluded from the removal. If the host fails the analysis, a warning dialog appears.

The following actions are also available:

- **Product Summary** gives additional information about the product or bundle and provides a **Product Description** button that displays information about additional information about dependencies, copyright, vendor, etc.

  The `Projected Action` column describes what type of removal is being done. The possible types are:

Remove	The product exists and will be removed.
Filesets Not Found	The system did not find the filesets as specified.
Skipped	The product will not be removed.
Excluded	The product will not be removed because of some analysis phase errors. See the logfile for details about the error.

  (The Product Summary List is not an object list. You cannot open the products, perform actions, or change the column view.)

- **Logfile** presents a scrollable view of detailed install information written to the logfile.
- **Re-analyze** repeats the analysis process.

Step IV: Removal

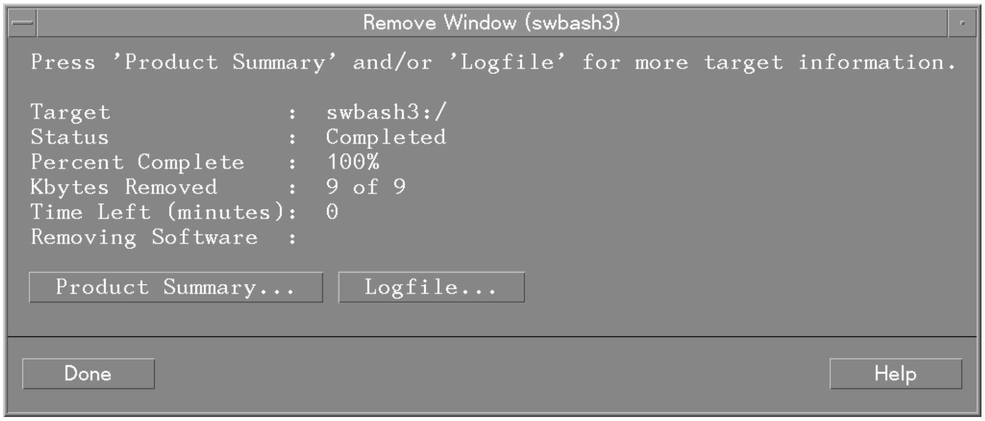
In this step, SD-UX proceeds with the actual removal.

After you click **OK** in the Analysis window, SD-UX starts removal and displays the Remove Window (Figure 3-4, “Remove Window,”), which shows status information.

These action buttons are available:

- **Done** returns you to the Software Selection Window. You can then begin another removal or exit the GUI (**File**→**Exit**).
- **Product Summary** display installation and product information (name, revision, installation results, installation summary).
- **Logfile** displays the logfile.

Figure 3-4 Remove Window



```
Press 'Product Summary' and/or 'Logfile' for more target information.

Target              :  swbash3:/
Status              :  Completed
Percent Complete    :  100%
Kbytes Removed      :  9 of 9
Time Left (minutes):  0
Removing Software   :
```

Removing with the Command Line

Syntax

```
swremove [XToolkit Option] [-d|-r] [-i] [-p] [-v]
[-C session_file] [-f software_file] [-Q date] [-s source]
[-S session_file] [-t target_file] [-x option=value]
[-X option_file] [software_selections] [@ target_selections]
```

Options and Operands

XToolkit Options	X window options for the GUI. See “XToolkit Options and Changing Display Fonts” on page 53.
`-d`	Operates on a depot rather than installed software. “Removing Software from Depots” on page 162 for more information.
`-i`	Runs a GUI or TUI interactive session. Used to “pre-specify” software selections for use in the GUI/TUI.
`-p`	Preview an install task by running it through the Analysis Phase and then exiting.
`-r`	Operate on an alternate root directory.
`-v`	Turn on verbose output to `stdout` and display all activity to the screen.
`-C` *`session_file`*	Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.
`-f` *`software_file`*	Read a list of software selections from a separate file instead of (or in addition to) the command line. See “Software Files” on page 58.
`-Q` *`date`*	Schedules a job for the given date when remote operations are enabled. See “Scheduling Jobs from the Command Line” on page 234 and Chapter 6, “Remote Operations Overview,” on page 189
`-S` *`session_file`*	Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-t` *`target_file`*
: Read a list of target selections from a separate file instead of (or in addition to) the command line. See “Target Files” on page 59.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Changing Command Options” on page 130.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 130.

`software_selections`
: The software objects to be removed. See “Software Selections” on page 56.

`target_selections`
: The target of the command. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swremove.

Table 3-9 **swremove Command Options and Default Values**

- admin_directory=/var/adm/sw
- agent_auto_exit=true
- agent_timeout_minutes=10000
- auto_kernel_build=true
- autoreboot=false
- autoremove_job=false
- autoselect_dependents=false
- autoselect_reference_bundles = true
- compress_index=false
- controller_source=
- distribution_target_directory= /var/spool/sw
- enforce_dependencies=true
- enforce_scripts=true
- force_single_target=false
- installed_software_catalog= products
- job_title=
- log_msgid=0
- logdetail=false
- logfile=/var/adm/sw/swremove.log
- loglevel=1
- mount_all_filesystems=true
- polling_interval=2
- preview=false
- remove_empty_depot=true
- reuse_short_job_numbers=true
- rpc_binding_info= ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- software=
- software_view=products
- targets=
- verbose=1
- write_remote_files=false

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

Remove Tasks and Examples

To remove a software product called MYSOFT from the default depot on the local host, type:

```
swremove -d MYSOFT
```

To preview the remove of the C and Pascal products installed at the local host:

```
swremove -p cc pascal
```

To remove a particular version of HP Omniback:

```
swremove Omniback,l=/opt/Omniback_v2.0
```

To remove the entire contents of a local depot:

```
swremove -d * @ /var/spool/sw
```

Removing Bundles

Removing a bundle does not always remove all filesets in that bundle. Because of SD-UX's dependency management features, a fileset that is required by another bundle will not be removed. For example, if the bundles `Pascal` and `FORTRAN` both use the fileset `Debugger.Run` and you try to remove `FORTRAN`, the fileset `Debugger.Run` will not be removed because it is also used by the bundle `Pascal`. This prevents the removal of one bundle from inadvertently causing the removal of a fileset needed by another bundle.

Removing Patches

You cannot remove patch software unless:

- Rollback files corresponding to the patch are available for re-installation.

— *or* —

- The base software modified by the patch is removed at the same time. (Removing the base software also removes the patches associated with that software.)

For more information on removing patches, see Chapter 5, "Managing Patches," on page 163.

Removing Multiple Versions

The swremove GUI can help simplify removal of multiple versions of a product.

Each separate version of a product along with its location directory is listed in the object list. Selecting a multiple version implies a `product:/location directory` pair. By default, the location is not displayed in the Software Selection Window. It can be displayed using the GUI **Columns Editor View→Columns...** menu item and enabling the **Product Directory** menu item.

You can select more than one version of a product during the selection phase. During analysis, a warning is generated if the version of the product exists on the target but at a different location. If the product exists on the target, it will be removed. If it does not exist on the target, the product is simply skipped. The **Product Summary...** button in the Remove Analysis Window gives a product-by-product summary of what will be removed if the remove phase is started.

(Multiple versions of products are inherently possible in a depot. No special handling or checks are required when removing from depots.)

Removing Software from an Alternate Root

Software can be removed relative to the primary root directory (`/`) or relative to an alternate root directory. An alternate root is a non-root location that can function as the root of a stand-alone system; that is, a system that can be unmounted and function as a self-contained system. Any information files used in software removal are retrieved from the Installed Product Database (see “Installed Products Database” on page 30) beneath this alternate root, not the IPD on the root volume.

4 Managing Software Depots

SD-UX uses software that is packaged and stored in a registered depot. This chapter discusses copying, listing, registering, removing, and verifying depot software.

Table 4-1 Chapter Topics

Topics:
"Depot Management Commands and Concepts" on page 134
"Copying Software Depots" on page 137
"Registering and Unregistering Depots (swreg)" on page 151
"Additional Depot Management Tasks and Examples" on page 155

Depot Management Commands and Concepts

The following commands will help you perform depot management tasks:

Table 4-2 Commands for Depot Management

Depot Task	Command(s)	More Information/Examples
Copy	swcopy	• "Copying Software Depots" on page 137 • "Combining Patch Depots" on page 155
Create	swcopy, swpackage	• Chapter 10, "Creating Software Packages," on page 301 • "Creating a Tape Depot for Distribution" on page 156
List	swlist -d	• "Listing Registered Depots" on page 158 • "Listing the Contents of a Depot (swlist -d)" on page 159
Register	swreg	• "Registering and Unregistering Depots (swreg)" on page 151
Remove	swremove -d	• "Removing Software from Depots" on page 162 • "Removing a Depot" on page 162
Unregister	swreg -u	• "Registering and Unregistering Depots (swreg)" on page 151
Verify	swverify -d	• "Verifying a Depot (swverify -d)" on page 161
Additional tasks		• "Additional Depot Management Tasks and Examples" on page 155

Depot Concepts

A **depot** is a special type of directory formatted for use by SD-UX commands, and used to contain software products. You can create a depot by using swcopy to copy software directly from physical media or by using swpackage to make a software package containing the depot.

When a depot resides on a networked system, that system can act as a source for software: other systems on the network can install software products from that server instead of installing them each time from media.

Network depots offers these advantages over installing directly from media:

- Several users can pull software down to their systems (over the network) without having to transport media to each user.
- Installation from a network server is faster than from media.
- You can combine different software products from multiple media or network servers into a single depot.

Types of Depots

A depot usually exists as a directory location. This software is in a hierarchy of subdirectories and filesets organized according to a specific media format. A host may contain several depots. For example, a software distribution server on your network might contain a depot of application software, a depot of patch software, and a depot of OS software.

There are two types of depots: directory and tape.

Directory Depot

- A directory depot consists of software stored under a special SD-UX-managed directory on your file system, usually `/var/spool/sw`.
- A directory depot can be writable or read-only.
- When you use the SD-UX commands to refer to a directory depot, you need only to refer to the depot's top-most directory. In a CD-ROM depot, this directory would be the CD-ROM mount point, such as `/cdrom/mydepot`.

Tape Depot

- Tape (serial) depots offer advantages when you must copy or install software over slow or unreliable network connections, including the web. (First copy the depot to a local host, then install from the local depot.)
- Software in a tape depot is formatted as a tar archive.
- Depots for actual cartridge, DAT and 9-track tape are referred to by the path to the tape drive's device file. For example: `/dev/rmt/0m`.
- You cannot modify or verify tape depots.
- You can create a tape depot only with the swpackage command. You cannot use swcopy to copy software directly to a tape. See Chapter 10, "Creating Software Packages," on page 301 for more information on swpackage.
- Software in a tape depot must first be transferred to a directory depot before it can be accessed by other hosts on the network.
- A tape depot can be accessed by only one command at a time.

Depot Registration

To make the software in a depot available for use by SD-UX commands across a network, you must register the depot. You can also unregister a depot if you do not want it to be available. See "Registering and Unregistering Depots (swreg)" on page 151 for more information.

Copying Software Depots

The swcopy command copies software between depots. *Software that is copied into a depot cannot be used directly*; it is placed there only to act as a source for installation and other SD-UX operations.

swcopy Features and Limitations

- swcopy does not perform compatibility checking.
- swcopy does not run control scripts.
- swcopy does not perform kernel building or rebooting, although it does perform other pre-install and postinstall checks, such as disk space analysis and requisite selection.
- When you create or modify a depot with swcopy, SD-UX automatically creates catalog files that describe the depot. These are stored in the IPD. See “Modifying the IPD (swmodify)” on page 115 for more information.
- Software dependencies apply to selections made with the swcopy GUI.

Using the swcopy GUI

Overview

This section provides an overview of the swcopy GUI.

- In general, all information presented in “Using the swcopy Command Line” on page 147 also applies to the swcopy GUI.
- This section also refers to information about standard GUI elements discussed in “Using the GUI and TUI Commands” on page 35.
- All information in this section also applies to the TUI program unless otherwise noted. See “The Terminal User Interface” on page 35.

The copy process has six steps:

Table 4-3 **Copy Process Steps**

I. Start-Up	Start the swcopy GUI.
II. Specify Target	Provide the location to which you want to copy the software.
III. Specify Source	Provide the location of the software depot from which the software will be copied.
IV. Select Software	Select the software you want to copy.
V. Analysis (Preview)	swcopy determines if the copy operation can succeed.
VI. Copy	The actual software copying process.

Step I: Start-Up

To start the GUI or TUI for an copy session, type:

```
/usr/sbin/swcopy
```

The GUI is automatically invoked *unless* you also specify software on the command line. To invoke the GUI *and* specify software, include the `-i` option. For example, to use the GUI for a preview (analysis only) session with `MyDepot`, type:

```
swinstall -i -p /MyDepot
```

The Software Selection window appears with the Specify Source dialog and the Select Target Depot Path dialogs superimposed over it.

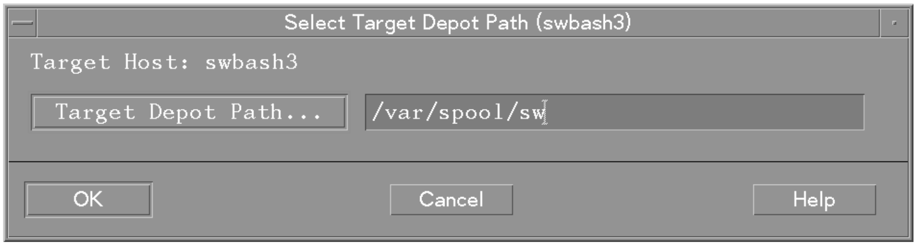
Step II: Specify Target

In this step, you specify the target to which SD-UX will copy the software.

(This step is skipped if you include the `-t target` option when you invoke the GUI. See “Using the swcopy Command Line” on page 147.)

The Select Target Depot Path dialog displays the default target depot. Since this matches the default source depot path, you must select a new target:

Figure 4-1 **Select Target Depot Path Dialog**



1. Enter a target path:
   - Type a new target path in the text box.

     — or —
   - Click the **Target Depot Path...** button. The Depot Paths dialog appears, listing registered depots on the host.

     a. Click on a depot in the list.

     b. Click **OK**. The Target Depot Path dialog disappears. The depot you selected is now displayed in the Select Target Depot Path dialog.
2. Click **OK**.

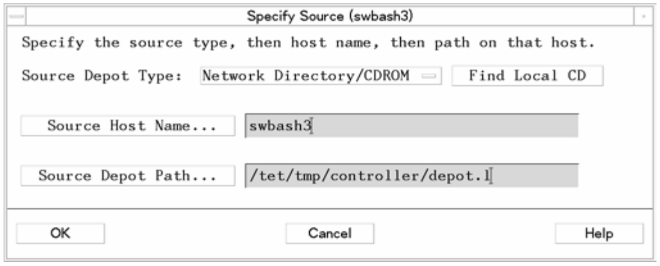
The Select Target Depot Path dialog disappears, and the Specify Source dialog is highlighted.

Step III: Specify Source

In this step, you must specify the source depot that contains the software you want to copy. The Specify Source dialog (Figure 4-2, “Specify Source Dialog,”) automatically lists the local host and default depot path.

(This step is skipped if you include the `-s source` option when you invoke the GUI. See “Using the swcopy Command Line” on page 147.)

Figure 4-2 Specify Source Dialog


Specify Source (swbash3)

Specify the source type, then host name, then path on that host.

Source Depot Type: Network Directory/CDROM

Source Host Name... swbash3

Source Depot Path... /tet/tmp/controller/depot.1

1. (Optional) To specify another host system, type a source host name, or:

   a. Click on the **Source Host Name** button. The system displays a dialog that lists all host system names contained in the `defaults.hosts` file (`$HOME/.sw/defaults.hosts` or `/var/adm/sw/defaults.hosts`).

   b. Choose a host name from the list.

   c. Click **OK**. The host name appears in the appropriate box in the Specify Source dialog.

2. (Optional) To specify the path to the depot, type a new path, or:

   a. Click on the **Source Depot Path** button to display a list of registered depots on the source host.

   b. Highlight one of the depots.

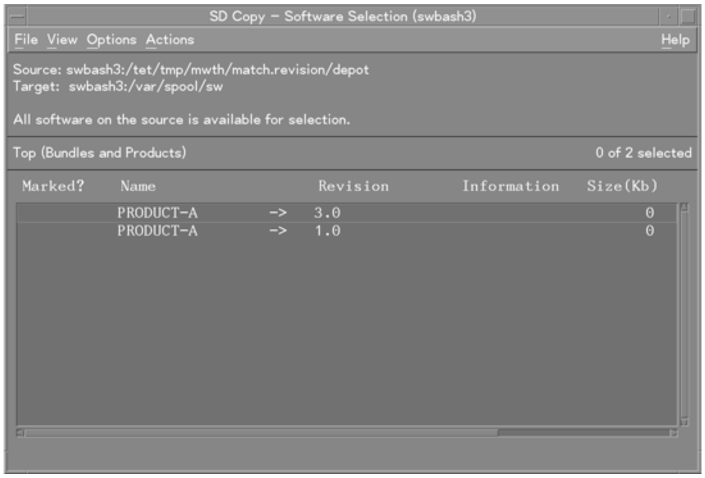
   c. Click **OK** to make it appear in the Specify Source dialog.

3. Click **OK**. The Specify Source dialog closes, and the Software Selection window displays the software contained in the depot you specified.

Step IV: Select Software

In this step, you use the Software Selection window (Figure 4-3, “Software Selection Window,”) to select the software you want to copy.

Figure 4-3 Software Selection Window


SD Copy – Software Selection (swbash3)
File View Options Actions
Help

```
Source: swbash3:/tet/tmp/mwth/match.revision/depot
Target:  swbash3:/var/spool/sw

All software on the source is available for selection.

Top (Bundles and Products)                                  0 of 2 selected

Marked?    Name                Revision        Information     Size(Kb)
           PRODUCT-A      ->   3.0                                    0
           PRODUCT-A      ->   1.0                                    0
```

1. Select software from the object list:
   a. Highlight an item
   b. Select **Actions→Mark For Copy**

      — *or* —

      Right-click to display the pop-up, then select **Mark For Copy**

   The `Marked?` flag in the object list changes to `Yes` to match your selection. (The flag `Partial` may appear if you select only some component of a software object.)

2. (Optional) Use additional choices from the **Actions** menu:
   - **Add Software Group** displays a list of previously saved software group files or lets you specify a directory. Selecting a file adds the software selections in the file to any selections you have already made in the Software Selection window.
   - **Save Software Group** lets you save your current list of marked software as a group.
   - **Manage Patch Selections** lets you select from a list of patches to copy, select filters for patches, and set other patch options. (See "Interactive Patch Management" on page 176 for more information.)
   - **Change Source...** cancels your software selections and returns you to the Specify Source dialog.
   - **Add New Codeword** lets you add a new codeword to unlock protected software. (This option is available only when SD-UX detects that the source contains protected software.)
   - **Show Description of Software** (available only for a single item highlighted in the object list) displays more information on the selected software.
   - **Change Target...** returns you to the Select Target Depot Path dialog ("Step II: Specify Target" on page 139).
3. Select **Actions**→**Copy** to start the analysis (preview) step. The Analysis dialog appears.

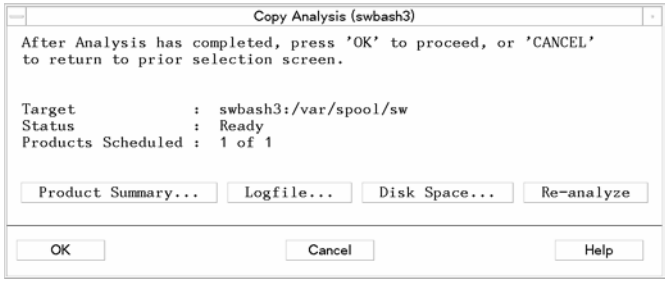
Step V: Analysis (Preview)

In this step, SD-UX analyzes the software you have selected.

The Analysis window displays status information about the analysis process. When the analysis is complete and the host status shows Ready, click **OK** to start the actual copy (see "Step VI: Copying" on page 146). The Analysis dialog is then replaced by the Copy dialog.

If you started a preview session, the copy stops after the analysis. Clicking **Cancel** returns you to the Software Selection window.

Figure 4-4 **Copy Analysis Dialog**



```
After Analysis has completed, press 'OK' to proceed, or 'CANCEL'
to return to prior selection screen.

Target            :  swbash3:/var/spool/sw
Status            :  Ready
Products Scheduled :  1 of 1
```

The following actions are available:

- **Product Summary** gives additional information about the product or bundle and provides a **Product Description** button that displays information about additional information about dependencies, copyright, vendor, etc.
- **Logfile** presents a scrollable view of detailed copy information written to the logfile.
- **Disk Space** displays the Disk Space Analysis window (Figure 4-5, "Disk Space Analysis Window,") which shows:
  - — The file system mount point,
  - — How much disk space was available before the copy,
  - — How much will be available after the copy,
  - — What percent of the disk's capacity will be used.
  - — How much space must be freed to complete the operation.

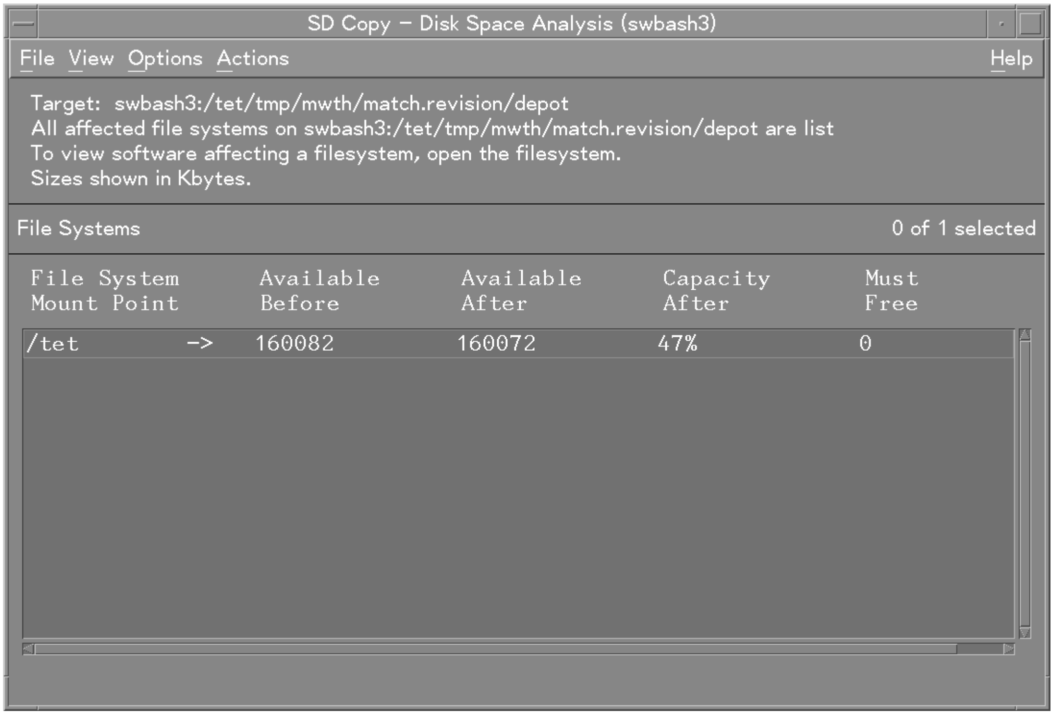
Menu choices in this window let you:

— Search the object list.

— Open items to look at the projected size requirements for specific filesets.

- **Re-analyze** repeats the analysis process.

Figure 4-5 Disk Space Analysis Window


SD Copy – Disk Space Analysis (swbash3)
File View Options Actions Help

Target: swbash3:/tet/tmp/mwth/match.revision/depot
All affected file systems on swbash3:/tet/tmp/mwth/match.revision/depot are list
To view software affecting a filesystem, open the filesystem.
Sizes shown in Kbytes.

File Systems 0 of 1 selected

```
File System      Available      Available      Capacity      Must
Mount Point      Before         After          After         Free
/tet         ->  160082         160072         47%           0
```

When Analysis completes, the status for any host displays as either `Ready` or `Excluded from task`. If *any* of the selected software can be copied onto the host, the status shows `Ready`. If *none* of the selected software can be copied onto the host, the status shows `Excluded from task`.

The following list summarizes the status results. You can find details about most problems by clicking the **Logfile** button.

`Ready` There were no errors or warnings during analysis. The copy may proceed without problems.

`Ready with Warnings`
Warnings were generated during the analysis. Errors and warnings are logged in the logfile.

`Ready with Errors`
At least one product selected will be copied. However, one or more products selected are excluded from the task because of analysis errors. Errors and warnings are logged in the logfile.

`Communication failure`
Contact or communication with the intended target or source has been lost.

`Excluded due to errors`
Some kind of global error has occurred. For example, the system might not be able to mount the file system.

`Disk Space Failure`
The copy will exceed the space available on the intended disk storage. For details, click the **Disk Space** button.

The `Products Scheduled` column shows the number of products ready for copying out of all products selected. These include:

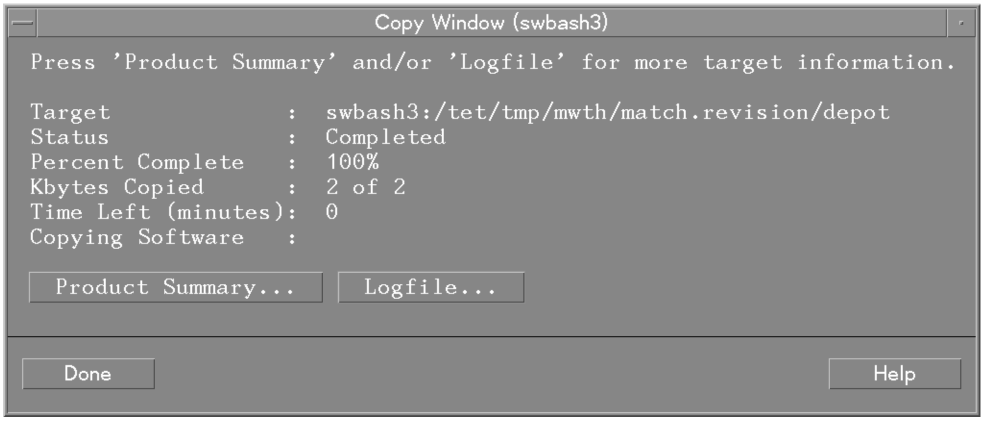
- Products selected only because of dependencies
- Partially selected products
- Other products and bundles that were selected

Step VI: Copying

In this step, SD-UX proceeds with the actual copy.

After you click **OK** in the Analysis window, SD-UX starts copying and displays the Copy Window (Figure 4-6, “Copy Window,”), which shows status information.

Figure 4-6 Copy Window



```
Press 'Product Summary' and/or 'Logfile' for more target information.

Target              :  swbash3:/tet/tmp/mwth/match.revision/depot
Status              :  Completed
Percent Complete    :  100%
Kbytes Copied       :  2 of 2
Time Left (minutes):  0
Copying Software    :
```

These action buttons are available:

- **Done** returns you to the Software Selection Window. You can then begin another copy or exit the GUI (**File**→**Exit**).
- **Product Summary** display copy and product information (name, revision, copy results, copy summary, product description).
- **Logfile** displays the logfile.

Using the swcopy Command Line

swcopy Syntax

swcopy [*XToolkit Options*] [-i] [-p] [-v] [-C *session_file*] [-f *software_file*] [-Q *date*] [-s *source*] [-S *session_file*] [-x *option=value*] [-X *option_file*] [*software_selections*] [@ *target_selections*]

Options and Operands

XToolkit Options X window options for the GUI. See “XToolkit Options and Changing Display Fonts” on page 53.

`-i` Run the GUI program. See “Using the swcopy GUI” on page 138.

`-p` Preview a copy task from the command line by running it through the Analysis Phase and then exiting.

`-v` Turn on verbose output to `stdout` and display all activity to the screen.

`-C` *session_file*
Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.

`-f` *software_file*
Read a list of software selections from a separate file instead of (or in addition to) the command line. See “Software Files” on page 58.

`-Q` *date* Schedules a job for the given date when remote operations are enabled. See “Scheduling Jobs from the Command Line” on page 234 and Chapter 6, “Remote Operations Overview,” on page 189

`-s` *source* Use the software source specified by *source* instead of the default, `/var/spool/sw`. The syntax is:

[*host*:][/*directory*]

host may be a host name, domain name, or internet address (for example, `15.1.48.23`). *directory* is an absolute path.

`-S` *session_file*
Run the command based on values saved from a previous session, as defined in *session_file*. See “Session Files” on page 61.

`-t` *`target_file`*
: Read a list of target selections from a separate file instead of (or in addition to) the command line. See "Target Files" on page 59.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See "Changing Command Options" on page 148.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See "Changing Command Options" on page 148.

`software_selections`
: The software objects to be copied. See "Software Selections" on page 56.

`target_selections`
: The target of the command. See "Target Selections" on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swcopy.

Table 4-4 swcopy Command Options and Default Values

- admin_directory=/var/adm/sw
- agent_auto_exit=true
- agent_timeout_minutes=10000
- allow_split_patches=false
- autoremove_job=false
- autoselect_dependencies=true
- autoselect_patches=true
- autoselect_reference_bundles=true
- codeword=
- compress_files=false
- compress_index=false
- controller_source=
- create_target_path=true
- customer_id=
- distribution_source_directory=/var/spool/sw
- distribution_target_directory=/var/spool/sw
- enforce_dependencies=true
- enforce_dsa=true
- job_title=
- layout_version=1.0
- log_msgid=0
- logdetail=false
- logfile=/var/adm/sw/swcopy.log
- loglevel=1
- max_targets=
- mount_all_filesystems=true
- patch_filter=software_specification
- patch_match_target=false
- polling_interval=2
- preview=false
- register_new_depot=true
- reinstall=false
- reinstall_files=true
- reinstall_files_use_cksum=true
- remove_obsolete_filesets=false
- retry_rpc=1
- retry_rpc_interval={0}
- reuse_short_job_numbers=true
- rpc_binding_info=
  ncacn_ip_tcp:[2121]
  ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=false
- select_local=true
- software=
- software_view=all_bundles
- source=
- source_cdrom=/SD_CDROM
- source_tape=/dev/rmt/0m
- source_type=directory
- targets=
- uncompress_files=false
- use_alternate_source=false
- verbose=1
- write_remote_files=true

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

Copy Tasks and Examples

This section provides examples of commands for copying software products. (See also “Additional Depot Management Tasks and Examples” on page 155.)

Simple swcopy Examples

To copy all products from the DAT tape at `/dev/rmt/0m` to the default depot (`/var/spool/sw`) on the local host:

```
swcopy -s /dev/rmt/0m \*
```

To copy a list of software selections (on a local CD-ROM) named in the file `mysoft` to a depot at the path `/depots/mydep/` on the host named *`hostA`* and preview the process before actually copying the software:

```
swcopy -p -f mysoft -s /mnt/cd @ hostA:/depots/mydep/
```

Using Software Codewords and Customer IDs

The swcopy command may prompt you for codewords if you try to access codeword protected software. You can also enter new codewords from the command line or from the GUI. This process is identical to that used by swinstall. See “Using Software Codewords and Customer IDs” on page 78 for more information.

Multiple Software Products in Depots

Software is packaged into products. Depots can store multiple versions of a product.

If a product version already exists in the depot, swcopy will *not* replace it unless the `reinstall` option is set to true. If this option is true, then the product is recopied.

If other versions of the product already exist in the depot, swcopy copies in the new version and the others are not changed.

swcopy does not automatically notify you when multiple versions of a product exist. swcopy notifies you only when an exact version exists. exists and will be skipped (or recopied)

Registering and Unregistering Depots (swreg)

To make the software in a depot available for use across a network by other SD-UX commands, you must **register** the depot. You can also **unregister** a depot if you do not want it to be available.

Depots are registered or unregistered in these ways:

- The swcopy command automatically registers newly created depots. (You can turn this function on or off with the `register_new_depot` option.)
- The swremove command automatically unregisters a depot after removing all the software the depot contains.
- The swreg command explicitly registers or unregisters depots.

The swreg command lets you explicitly register or unregister depots when the automatic registration features of swcopy or swremove are not enough. For example, you can use swreg to:

- Make a CD-ROM or other media available as a registered depot.
- Register a depot that was created with swpackage.
- Unregister a depot to restrict network access without physically removing the depot from a host.

Register Media or Create Network Depot?

When does it make sense to use your software media as a registered depot versus using the media to create a network depot? In general, using media as a depot makes sense for small-scale use, such as when only one or two other systems need to access the media. If more systems will need to access the media, performance will be better if you create a network depot from the individual media. See “Additional Depot Management Tasks and Examples” on page 155 for an example.

Registration and Security

Because SD-UX stores its objects in the file system, someone could build a “Trojan Horse” file system image of a software depot. This could breech the security of any system that installed products from the false depot. To protect systems from such a situation, SD-UX requires that depots be registered before software may be installed or copied from it. This check is always performed before granting access, except when swinstall is run by the local superuser.

NOTE Registration of a depot does not enforce any access restrictions. Access enforcement is left to SD security (see Chapter 9, “SD-UX Security,” on page 255). Registration with swreg requires insert permission in the host’s ACL.

Authorization

To register a new depot or to unregister an existing depot, swreg requires read permission on the depot in question and insert permission on the host. To unregister a registered depot, the swreg command requires write permission on the host. See Chapter 9, “SD-UX Security,” on page 255 for more information on permissions.

Using swreg

swreg Syntax

`swreg -l` *`level`* `[-u] [-v] [-C` *`session_file`*`] [-f` *`object_file`*`]`
`[-S` *`session_file`*`] [-t` *`target_file`*`] [-x` *`option=value`*`]`
`[-X` *`option_file`*`] [`*`objects_to_register`*`] [@` *`target_selections`*`]`

Options and Operands

`-l` *`level`*	Specifies the level of the object to register or unregister, where level can be `depot` or `root`.
`-u`	Causes swreg to unregister the specified objects instead of registering them.
`-v`	Turns on verbose output to `stdout` and displays all activity to the screen.

`-C` *`session_file`*
: Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.

`-f` *`object_file`*
: Reads a list of depots or root objects to register or unregister from a *`object_file`* instead of (or in addition to) the command line.

`-S` *`session_file`*
: Run the command based on values saved from a previous session, as defined in *`session_file`*. See “Session Files” on page 61.

`-t` *`target_file`*
: Read a list of target selections from a separate file instead of (or in addition to) the command line. See “Target Files” on page 59.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Using Command Options” on page 59.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Using Command Options” on page 59.

`objects_to_register`
: The software objects to be registered or unregistered.

`@`*`target_selections`*
: The target on which the objects will be registered or unregistered. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swreg:

Table 4-5 swreg Command Options and Default Values

- admin_directory=/var/adm/sw
- distribution_target_directory= /var/spool/sw
- level=
- log_msgid=0
- logfile=/var/adm/sw/swreg.log
- logdetail=false
- loglevel=1
- objects_to_register=
- rcp_binding_info= ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- select_local=true
- targets=
- verbose=1

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

swreg Examples

To unregister a CD-ROM depot mounted at `/mnt/cd`, you would type:

```
swreg -l depot -u /mnt/cd
```

To register the same depot (mounted at `/mnt/cd` on the local host) as a depot to be available on the network, type:

```
swreg -l depot /mnt/cd
```

The following example enables direct access from one or two other systems to the `HWEnable11i` depot on the Support Plus CD, assuming the Support Plus CD is mounted at `/cdrom`:

```
swreg -l depot /cdrom/HWEnable11i
```

Additional Depot Management Tasks and Examples

This section illustrates some typical depot management tasks and provides extended examples of how you can use SD-UX to manage your environment.

Combining Patch Depots

This example shows how to combine into a single depot five downloaded patches (which are tape depots) from HP. The example also shows how to register the depot, list the depot contents, and install the patches from the new depot using the `patch_match_target` option. The example assumes that you have already downloaded patches PHKL_20349, PHKL_22161, PHSS_21906, PHSS_21950, and PHCO_22923 from the HP ITRC (**`http://itrc.hp.com/`**):

```
swcopy -s /tmp/PHKL_20349.depot \* @ /depots/mypatches

swcopy -s /tmp/PHKL_22161.depot \* @ /depots/mypatches

swcopy -s /tmp/PHSS_21906.depot \* @ /depots/mypatches

swcopy -s /tmp/PHSS_21950.depot \* @ /depots/mypatches

swcopy -s /tmp/PHCO_22923.depot \* @ /depots/mypatches

swreg -l depot @ /depots/mypatches

swlist -d -s /depots/mypatches

swinstall -x patch_match_target=true -s /depots/mypatches
```

Creating a Tape Depot for Distribution

This example shows you how to create a tape depot as a single file that can be distributed via ftp or the web. This example uses the five patches from the previous example (which are formatted as tape depots) and uses an existing depot at `/depots/mypatches`. The swlist command shows the depot contents (see “Listing the Contents of a Depot (swlist -d)” on page 159).

```
swpackage -x media_type=tape -s /depots/mypatches \
           @ /tmp/mypatches.depot
```

```
swlist -d -s /tmp/mypatches.depot
```

To create a tape depot from `myproduct.psf`, a valid product specification file:

```
swpackage -x media_type=tape -s myproduct.psf \
          @ /tmp/myproduct.depot
```

```
swlist -d -s /tmp/myproduct.depot
```

See Chapter 10, “Creating Software Packages,” on page 301 for more information about swpackage.

Setting Depot Attributes

When you create a depot, you may want to set the title and description attributes to help identify the depot and what it contains.

At the top of the product specification file (psf) for the depot, place the lines similar to the following:

```
distribution
title Optional Development Tools
description "Text processing and programming tools
      \generally useful for source code development"
```

Then package and register the depot:

```
swpackage -s mydepot.psf @ /depots/mydepot
```

```
swreg -l depot @ /depots/mydepot
```

To see the title and description of all depots on a system:

```
swlist -v -a title -a description -l depot
```

Creating a Network Depot

Creating a network depot from which to install software can improve performance and ease of use when you have to install software to large numbers of systems. For example, HP-UX 11i is delivered on two CDs, requiring you to swap CDs during the update process. To perform an update without having to swap CDs, you can create a remote depot on an existing 11i system that contains all the necessary software, then update from that single source. (For more information on the update process, see the *HP-UX 11i Installation and Update Guide*.)

As root, follow this procedure to create a network depot from the HP-UX 11i CDs onto a depot server or other system running HP-UX 11i in your network:

1. Verify that you have at least 1,230 MB of free space to create the network depot on another system in your network. If this space is not available, use SAM to either create a new volume group or extend an existing volume group. For help, see either SAM help or the *Managing Systems and Workgroups* manual.
2. Login as root and mount the logical volume on a new directory named */update*. This directory will hold your network depot.
3. Insert the HP-UX 11i CD1 and wait for the CD drive's busy light to stop blinking.
4. Find the CD-ROM device file name:

   ```
   ioscan -fn | more
   ```

   A typical CD-ROM device name is: `/dev/dsk/c1t2d0`
5. Create the directory */cdrom* under root(`/`):

   ```
   mkdir /cdrom
   ```
6. Mount the CD onto the */cdrom* directory as a file system. For example:

   ```
   mount /dev/dsk/c1t2d0 /cdrom
   ```
7. Merge all products on the mounted CD to the target depot, for example */update/update-depot*:

   ```
   swcopy -s /cdrom \* @ /update/update-depot
   ```
8. Unmount the CD from directory */cdrom*:

   ```
   umount /cdrom
   ```

9. Insert the HP-UX 11i CD2. Wait for the drive's busy light to stop blinking.
10. Repeat Steps 6 through 8 using CD2 and the Support Plus CD.

The network depot is now ready for you to use to update your HP-UX 10.20 or 11.0 system to HP-UX 11i.

Managing Multiple Versions of HP-UX

You can use your HP-UX 11i system to manage depots for HP-UX 11.00 and 10.20, with the following guidelines:

- HP recommends that you do not mix OS versions within the same depot. That is, locate 10.20 software in 10.20 depots, 11.00 software in 11.00 depots, and 11i software in 11i depots.
- You can manage 11.00 depots from 11i without any special considerations—although you should maintain the segregation of 11.00 and 11i software in separate depots. Although the formats are similar, 11i software may contain vendor-defined attributes not recognized by 11.00 systems. This results in warnings when 11.00 systems access 11i software.
- For 10.20 depots:
  - To create 10.20 depots from an 11i system, you must use the correct layout_version. For example:

    ```
    swcopy -x layout_version=0.8 ...
    ```

    ```
    swpackage -x layout_version=0.8 ...
    ```

    From then on, your 11i system can maintain the 10.20 depot.
  - SD-UX will generate warnings if you attempt to put `layout_version=1.0` software (11.00 or 11i format) into a `layout_version=0.8` (10.20) depot.

Listing Registered Depots

swlist can display lists of registered depots residing on a host. To do this, use combinations of the `-l depot` option.

To list all depots on the local host, type:

```
swlist -l depot
```

To list all depots on a remote machine (`hostA`), type:

```
swlist -l depot @ hostA
```

To list all the depots on a system from newest to oldest (by time last modified):

```
swlist -l depot -a mod_date -a mod_time | sort -rn -k 7,7
```

TIP Use the `mod_time` as a convenient sort field (a single integer), and use `mod_date` to include human-readable output. (Place `mod_time` at the end of the display where it's less visible.)

Listing the Contents of a Depot (swlist -d)

With swlist you can list all software that is packaged, stored, and ready to be installed.

The `swlist -d` option lets you list software residing on the default depot on your local host. For browsing any depot in the GUI, you can use `swlist -i -d`. You can also view the associated session and audit log files.

NOTE By default the output of swlist will reflect the POSIX format for attributes. This may affect users who parse this output.

In the following examples, swlist output requests are sent to standard output. All examples assume the `one_liner` option is `revision size title` and the `level` option is `product` or undefined.

List the contents of the local tape depot, `/dev/rmt/0`:

```
swlist -d @ /dev/rmt/0
```

```
AUDIT     3.5   9834    Trusted Systems Auditing Utils
COMMAND   1.7   4509    Core Command Set
C-LANG    2.5   5678    C Programming Language
DISKLES   1.8   6745    HP Cluster Commands
KERNEL    1.4   56908   Kernel Libraries and Headers
VUE       1.3   5489    Vue (Instant Ignition Release)
WINDOWS   2.06 10423  Windowing Products
```

List the media attributes of the local tape depot, `/dev/rmt/0`:

`swlist -d -v -l depot @ /dev/rmt/0`

```
type        distribution
tag         CORE OS
description HP-UX Core Operating System Software Disc
number      B2358-13601
date       June 1991
```

List the products stored in the software depot on `host1` located at `/swmedia`. For this example assume `one_liner` is `title size architecture`:

`swlist -d @ host1:/swmedia`

```
FRAME Frame Document Pkg 2319 HP-UX_9000_Series700/800_AorB
FRAME Frame Document Pkg 2458 OSF1_9000_Series700_1.0
ME30  3-D Mechanical Eng 5698 HP-UX_9000_Series300/800_AorB
SOFTBENCH Softbench Development Env 4578 HP-UX_9000_Series30
0
TEAMWORK Tmwk. Design/Analysis 3478 HP-UX_9000_Series300/400
```

Note that the media contains two versions of the FRAME product.

Source Depot Auditing

If both the source and target systems are 10.30 or later versions of HP-UX, you can use `swlist` to audit the depot. The system administrator at the source depot machine can turn the audit functionality on or off. This feature tracks users and their software selections. In addition, you can determine when depots are being used.

As the administrator, you must set to true the value of `swagent.source_depot_audit` in the `/var/adm/sw/defaults` file for swagent. This creates a `swaudit.log` file on the source depot (for writable directory depots) or in `/var/tmp` (for tar image, CD-ROM, or other non-writable depots). This works like `swagent.log` for source depot.

You can view the audit files by typing **`swlist -i -d`**. As long as the system has the corresponding SD message catalog files on it, you can view the audit information on a remote/local depot (with your language preference set).

Verifying a Depot (swverify -d)

To can use the swverify command to verify the software within a depot. swverify performs these tasks:

- Verifies that all dependencies (prerequisites or corequisites) can be met.
- Reports missing files.
- Checks file attributes, including permissions, file types, size, checksum, mtime, and major/minor attributes.

For example, to verify the entire contents of a local depot:

```
swverify -d \* @ /var/spool/sw
```

NOTE The swverify command does not execute vendor-supplied verification scripts within a depot.

Removing Software from Depots

Invoking swremove with the -d option removes software from depots instead of root file systems. This also means that you must specify a path to identify the depot from which you want to remove the software. For example:

```
swremove -d Old-Software @ /var/spool/sw
```

For the `swremove -d` GUI, you are prompted to specify the depot by a dialog that appears after you invoke the GUI. This is the same dialog used to specify a depot target for swcopy operations. See “Step II: Specify Target” on page 139 for information about how to use this dialog.

Removing a Depot

To remove and automatically unregister a depot:

```
swremove -d \* @ /tmp/MyDepot
```

5 Managing Patches

This chapter discusses Software Distributor features that help you develop, install, and manage software patches.

Table 5-1 Chapter Topics

Introduction

SD-UX gives you the ability to perform patch management operations including installation, copying, listing, removing, rollback, and committal. Patch-related features of SD-UX include:

- Command options for patch management functions.
- Software objects and attributes for identifying and managing patches.
- An interactive patch management tool.

NOTE You can use the SD-UX remote operations features for patch management. See Chapter 6, “Remote Operations Overview,” on page 189 and “Installing Patches to Remote Systems” on page 174.

For More Information For more information on patching HP-UX products, see *HP-UX Patch Management: A Guide to Patching HP-UX 11.X Systems*. This document is available under the **Patch Management** section of:

`http://docs.fc.hp.com/hpux/os/11i/oe/`

Patch Concepts

A patch is defined as software packaged with the `is_patch` attribute set to true.

As with non-patch software, patches are structured into products and filesets. By convention on HP-UX, patch products are given unique names, but their fileset names match the corresponding base filesets that they patch. In general, patches are intended to be managed (that is, installed, copied, or removed) at the product level.

Each patch fileset has associated with it an **ancestor** fileset, which is the base software that it patches. A patch fileset may not be installed on a target system unless its ancestor fileset is also being installed or is already present on the system. Similarly, an ancestor fileset cannot be removed without also removing all of its patches. A patch fileset’s ancestor is identified by its `ancestor` attribute.

Patches that have been applied to an ancestor fileset are listed in the ancestor's `applied_patches` attribute.

HP patches are required to completely replace earlier patches. A newer version of a patch is said to **supersede** an earlier version. A patch fileset's `supersedes` attribute lists all previous patch filesets that it supersedes.

Patch filesets can have dependencies on other patches as well as non-patch software. When a patch supersedes another patch, it is also assumed to be able to satisfy any dependencies on that earlier patch. (See "Patch Supersession and Dependency Resolution" on page 166.)

By default, patches installed on a target system can be rolled back, that is, the files that the patch replaced are stored in a special save area so they can be restored if you remove the patch later on.

A patch that has been installed on a target system is assigned a `patch_state` attribute value that indicates whether it can be rolled back and whether it has been superseded.

Patches can be selected and managed explicitly, or automatically as part of the selection of non-patch software.

You can manage patches separately from regular software items. Selection for installation and listing is supported by the `category_tag` attribute and special patch management options to SD commands.

See "Packaging Patch Software" on page 182 in this chapter for complete information on patch-related attributes and objects.

Patch Installation Paradigm

On HP-UX 10.x, SD-UX did not distinguish patch software from non-patch software. The `match_target` command option was used to select the appropriate patches for software present on the target.

This paradigm has changed for HP-UX 11.x with the addition of new functionality:

- SD now distinguishes patches from non-patch software based on new, patch-specific software attributes.
- The `match_target` option is still supported, but is now used only for the selection of non-patch software. A new option, `patch_match_target`, is used to select patches that correspond to the software already present on a target depot or root.

- Another new option, `autoselect_patches`, causes SD to automatically select patches that are appropriate for any non-patch software that has been selected.
- Patch dependencies are now enforced. See "Patch Supersession and Dependency Resolution" on page 166 and "Using swlist to Resolve Manual Dependencies" on page 179.

Patch management command options are discussed in the following sections.

Patch Supersession and Dependency Resolution

Because patches can be superseded, special considerations are made when resolving dependencies on patches. By definition, a patch that supersedes another patch is assumed to be able to satisfy any dependencies that exist on the superseded patch. The converse is not true, however. A superseded patch will not satisfy a dependency on a superseding patch.

SD-UX uses the information in each patch fileset's `supersedes` attribute to build a **supersession chain**, which represents the superseding relationships among patches for a given base fileset.

When resolving a dependency on a patch, SD-UX attempts to find the latest patch in the corresponding supersession chain. It starts with the patch specified by the dependency, and traverses the chain until it finds a non-superseded patch. Thus, a dependency that specifies a given patch may actually cause a different, superseding patch to be automatically selected.

You can override this auto-selection behavior by explicitly selecting patches. If SD-UX encounters an explicitly-selected patch fileset during the traversal of the supersession chain, that patch is used to resolve the dependency—even if it is superseded by other patch filesets in the chain.

Note, however, that if the chain has an explicitly-selected patch fileset that cannot satisfy the dependency (that is, it is superseded by the patch fileset specified in the dependency), swinstall will treat this as an error (an attempt to install multiple patch filesets in the same supersession chain).

Patch-Related Features

SD's patch-related features include command options and software attributes. Patch attributes are discussed in "Packaging Patch Software" on page 182 in this chapter.

Command Options

Patch default options are available at the command line. You can change their default values by:

- Specifying values with the `-x` command-line option
- Changing the default options files
- Using the swinstall or swcopy GUI via the **Actions→Manage Patch Selection...** choice.

For complete information on default options, see "Using Command Options" on page 59 and Appendix A, "Command Options," on page 421.

The following table summarizes by command the SD-UX default options for managing patches:

Table 5-2 **Patch Options Listed by Corresponding Command**

Command	Patch Option

`swask`	`autoselect_patches=true`
	`patch_filter=*.*`

`swcopy`	`allow_split_patches=false`
	`autoselect_patches=true`
	`patch_filter=*.*`
	`patch_match_target=false`

`swinstall`	`allow_split_patches=false`
	`autoselect_patches=true`
	`patch_filter=*.*`
	`patch_match_target=false`
	`patch_save_files=true`

Table 5-2 Patch Options Listed by Corresponding Command (Continued)

Command	Patch Option

`swlist`	`level=patch` (equivalent to `-l patch`)
	`patch_one_liner=title patch state`
	`show_superseded_patches=false`

`swmodify`	`patch_commit=false`
`swremove`	`allow_split_patches=false`

- **allow_split_patches=false**

  Permits the independent management of individual patch filesets within a patch product. This option should be used only to resolve critical problems when directed by your HP support representative.

  Applies to swinstall, swcopy, and swremove.

- **autoselect_patches=true**

  Automatically selects the latest patches (if any) for a software object that you have selected for a swinstall or swcopy operation. (Selection is based on the patch object's `supersedes` and `ancestor` attributes.) The default value is true. The patches must reside in the same depot as the selected software.

  This option is useful for installing patches at the same time that you install the base software to which the patches apply.

  You can use this option with the `patch_filter` option to limit your automatic patch selection.

  Applies to swask, swinstall, and swcopy.

- **level=**

  Controls the depth of swlist output. When set to the value `patch`, swlist lists patches and their `patch_state` (`applied`, `committed`, `superseded`, or `committed/superseded`) for each ancestor fileset. This option is equivalent to the `-l patch` command-line option.

  Applies to swlist only.

- **patch_commit=false**

  Commits a patch by removing files saved for patch rollback. The default value is false. When set to true, this option removes the saved files for the patches specified in the software selections for the command and changes the associated `patch_state` attribute from `applied` to `committed` or from `superseded` to `committed/superseded`. (See also `patch_save_files`.)

  Note that when a patch is committed, all patches that it has superseded are also committed.

  Applies only to swmodify only.

- **patch_filter=*.***

  Specifies a filter used during the automatic patch selection process.

  This option is used in conjunction with the `autoselect_patches` and `patch_match_target` options to filter out patches that do not meet the specified criteria (tag, version, etc.).

  Applies to swask, swcopy and swinstall.

- **patch_match_target=false**

  Automatically selects the latest patches that correspond to software on the target root or depot. The default value is false.

  This option is useful when you are installing patches on previously installed software.

  This option can be used with the `patch_filter` option to filter out patches that do not meet the specified criteria.

  Applies to swcopy and swinstall.

NOTE When you use the SD-UX remote operations features to push patches to remote systems, you can use `patch_match_target` with only one remote system at a time.

- **patch_one_liner=title patch_state**

  Specifies the attributes shown on the one-line display for each object listed by swlist. Applies when the `-l patch` option is invoked and when no `-a` or `-v` option is specified. The default display attributes are `title` and `patch_state`.

  Applies to swlist only.

- **patch_save_files=true**

  Saves files to be patched before they are overwritten during an installation. This permits future rollback of patches. When set to false, patches cannot are automatically committed and can not be rolled back (removed) unless the base (non-patch) software modified by the patch is removed at the same time. The default value is true.

  Applies to swinstall only.

- **show_superseded_patches=false**

  Controls the swlist display of superseded patches. By default, swlist will not display superseded patches, even if you explicitly list the superseded patch. To view superseded patches, set this option to true.

  Applies to swlist.

Patch Management Tasks and Examples

installing copying, interactive patch management, removal, rollback, committing patches, verifying patches

Installing Patches

Installation of patch products follows the same rules as any other SD installation. The key difference is that patch selection and mechanisms let you select only the patches that meet specified criteria. Selection mechanisms for patches are:

- The `category_tag` attribute and corresponding `category` objects.
- The `patch_filter`, `patch_match_target`, and `autoselect_patches` options.

When you install a patch, SD updates the `applied_patches` attribute of the fileset that has been patched and updates the `INFO` file information to include the patched file’s attributes. Also (if the `patch_save_files` option is set to true), files that would be overwritten are stored in a special save area in the IPD.

When a patch is installed, by default it has the `patch_state` of applied. When the patch is committed (rollback files are removed) or it has been installed without saving rollback files, it has the state of committed. When the patch is superseded, the `patch_state` is set to superseded, and the `superseded_by` attribute is set to the *`software_specification`* of the superseding patch fileset.

If a patch is both committed and superseded, its `patch_state` is set to committed/superseded.

Installing Patches in Same Session as Base Product

If you select a non-patch fileset for installation and patch filesets for that base fileset exist in the same source depot, all applicable patches are selected by default as long as the `autoselect_patches` option is set to its default value of true. The following rules also apply:

- Automatic patch software selections are filtered as defined by the `patch_filter` option.

- If more than one patch for a base fileset exists, only the latest patches (i.e., those that are not superseded) will be automatically selected for installation, unless overridden by an explicit patch selection.
- You can also explicitly specify patches on the command line. See “Explicitly Specifying Patches” on page 173 for more information.

The following examples demonstrate the use of patch options on the command line. (Note that the `autoselect_patches` option is true by default.)

The example below shows the default behavior for patch installation. All patches in the depot that apply to the software being installed (in this case, `X11`) are selected by default:

```
swinstall -s sw_server X11
```

To select all applicable patches that include the `category_tag` of `critical_patch` and install them along with the selected software:

```
swinstall -s sw_server \
          -x patch_filter=”*.*,c=critical_patch” X11
```

The following example installs a product and an explicitly specified patch.

```
swinstall -s sw_server \
          -x autoselect_patches=false X11 PHSS_12345
```

Installing Patches After Base Product Installation

When you want to install patches after installation of the base product, you can select the patches explicitly or by matching the installed software using the `patch_match_target` option, which automatically selects the latest patches for the software found on the target.

To select all patches in the depot that correspond to currently installed software:

```
swinstall -s sw_server -x patch_match_target=true
```

To select all patches in the depot that correspond to currently installed software and that contain the `category_tag` critical_patch:

```
swinstall -s sw_server -x patch_match_target=true \
          -x patch_filter=” *.*,c=critical_patch”
```

Patch Filtering with Multiple Criteria

You can repeat a version qualifier (for AND criteria) and use the pipe symbol (`|`) within qualifiers (for OR criteria). This is consistent with the current level of expression support in POSIX standard software specifications.

To install any patches that have the category tag of `critical` AND the category tag of either `special_release` OR `hardware_enablement`.

```
swinstall -s sw_server -x patch_match_target=true \
          -x patch_filter=”*.*,c=critical,\
           c=special_release|hardware_enablement”
```

NOTE Selecting software with the `\*` wildcard overrides patch filtering.

Explicitly Specifying Patches

You can explicitly specify and install a patch (without autoselection or matching the target) by specifying one or more operands for the *`software_specification`* within a command.

Explicit patch selections override any automatic patch selections. When SD automatically selects a patch (for example, with a value of true for the `auto_select_patches` option), it attempts to select the latest patch in the supersession chain unless some other patch in the chain is explicitly selected.

To explicitly install a patch:

```
swinstall PHCO_1234
```

NOTE Patch filtering does not apply to explicitly selected patches.

Installing Patches to Kernel and Library Files

To permit patching of kernel files or libraries (e.g. `libc.a`), SD uses an `archive` file type of `a`. When loading a file of type `a`, swinstall temporarily installs the `.o` file to the target path specified, integrates it into the archive specified by the `archive_path` attribute of the file, and then removes the `.o` file.

If patch rollback is enabled (see “Patch Removal, Rollback, and Committal” on page 179), the original `.o` file is automatically extracted first and saved so that it can be replaced. Disk Space Analysis is performed as needed to account for these operations.

Patch Load Order

If you install patch filesets and normal filesets in the same session, then each patch fileset is considered to have an implied prerequisite on the fileset that it is patching. For example, a product containing the patch fileset is installed (or copied into serial distributions) *after* installation of the one or more products that contain the patch’s ancestors.

If a base fileset has the `is_kernel` attribute set to true, then the fileset patching it must also have the `is_kernel` attribute set to true to be installed in the kernel phase of the execution. Otherwise, the patch is installed along with other non-kernel filesets.

If a bundle contains both normal and patch filesets, the filesets are installed in their normal order except that any ancestor fileset must be installed before its patch or patches.

Updating Patched Software

Installation of a new version of a base fileset results in removal of all filesets that patch the base fileset that you are replacing, along with any files saved for potential rollback.

Installing Patches to Remote Systems

You can use the SD-UX remote operations features to install patches on multiple remote systems. (See Chapter 6, “Remote Operations Overview,” on page 189.) You can explicitly select patches for multiple remote systems. Note, however, that the `patch_match_target` option works with only one remote system at a time.

Copying Patches

The swcopy command uses the `autoselect_patches`, `patch_filter`, and `patch_match_target` options in the same way that swinstall does, except that there is no filtering based on architecture (either 32-bit or 64-bit).

The following example copies `X11` software from the default depot and copies all patches for this software at the same time. (Note that `autoselect_patches` is true by default.)

```
swcopy X11 @ hostA:/tmp/sw
```

The following example copies patches that match the base filesets that are already present in the target depot, and copies (at the same time and from the same depot) a filtered set of patches (which have a category_tag value of `hardware_enablement`) for the base software being copied. (Note that `autoselect_patches` is true by default.)

```
swcopy -x patch_filter=”*.*,c=hardware_enablement” \
       -x patch_match_target @ hostB:/tmp/newdepot
```

To copy all patches for the base filesets that are already present in the target depot, starting from a depot that contains patch and non-patch software:

```
swcopy -x patch_match_target=true \
        @ hostC:/var/spool/sw
```

To copy a filtered set of patches for the base filesets that are already present in the target depot, starting from a depot that contains patches and that may contain non-patch software:

```
swcopy -x patch_match_target=true \
       -x patch_filter=”*.*,c=special_release” \
        @ hostD:/var/spool/sw/sample.depot
```

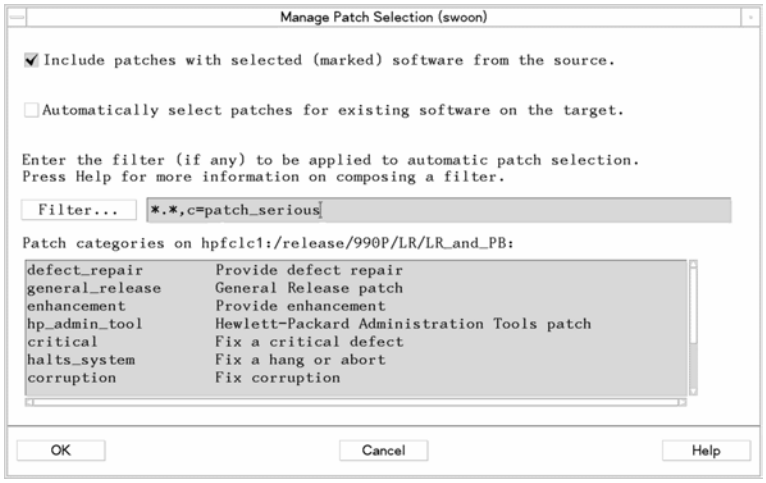
Interactive Patch Management

The swinstall or swcopy GUI lets you perform interactive patch installation and copying. (See “Installing with the GUI” on page 65 and “Using the swcopy GUI” on page 138.)

The **Manage Patch Selection...** option in the **Actions** menu opens the **Manage Patch Selection** dialog. This dialog lets you:

- Select from a list of patches available to install or copy.
- Select filters for patches.
- Set other patch options.

Figure 5-1 Manage Patch Selection Dialog



✓ Include patches with selected (marked) software from the source.

Automatically select patches for existing software on the target.

Enter the filter (if any) to be applied to automatic patch selection.
Press Help for more information on composing a filter.

Filter... `*.*,c=patch_serious`

Patch categories on hpfclc1:/release/990P/LR/LR_and_PB:

```
defect_repair       Provide defect repair
general_release     General Release patch
enhancement         Provide enhancement
hp_admin_tool       Hewlett-Packard Administration Tools patch
critical            Fix a critical defect
halts_system        Fix a hang or abort
corruption          Fix corruption
```

The main object list contains a read-only list of available patch categories. The list contains the name of the category and a short description. You can use the list as an aid to selecting and filtering patches. The following options are also available:

- **Include patches with selected (marked) software from the source**

  Sets the `autoselect_patches` option. (Default is true.)

- **Automatically select patches for existing software on the target**

  Sets the `patch_match_target` option. (Default is false.)

- **Filter** (button and specification field)

  Click on the **Filter** button to display a list of example filters that you can select from. (To change the filters in the list, see “Editing the Patch Filter List” on page 177.)

  This field sets the `patch_filter` option, which lets you specify a filter for automatic patch selection. Patches for software to be installed or copied are automatically marked as you enter the analysis phase. Only patches that match the filter criteria are marked.

You can also set **Save files replaced by patch for later rollback** in the **Options** menu. This sets the `patch_save_files` option. (The default is true.)

See “Command Options” on page 167 in this chapter for more information on patch options.

NOTE As with all system options, the patch management options revert to their default values at the next session unless you save and re-use the session information. See “Session Files” on page 61 for more information.

Editing the Patch Filter List

You can change the default list of patch filters displayed by the swinstall and swcopy GUI. The list is stored in:

- `/var/adm/sw/defaults.patchfilters`

  The system-wide default list of patch filters.

- `$HOME/.swdefaults.patchfilters`

  The user-specific default list of patch filters.

The list of patch filters is enclosed in braces {} and separated by white space (blank, tab, or newline). For example:

```
swinstall.patch_filter_choices={
*.*,c=enhancement
*.*,c=critical
}
swcopy.patch_filter_choices={
*.*,c=halts_system
}
```

Listing Patches

Software objects with the `is_patch` attribute set to true have the built-in, reserved category of `patch`. This lets you list available patches and patches with a certain name.

You can also list patches with the swlist GUI (invoked by **`swlist -i`**).

For example, to list all products and bundles in a depot that have the `is_patch` attribute set to true:

```
swlist -d -l product -l bundle *,c=patch
```

```
...
PHSS_15851  1.0  Xserver cumulative patch
PHSS_16482  1.0  CDE Localization for UTF8 locales
PHSS_16587  1.0  HP aC++ runtime libraries
...
```

You can list the patches that have been installed for a given base software product or fileset using the `-l patch` option.

For example, to list the patches applied to the `X11.X11R6-SHLIBS` fileset:

```
swlist -l patch X11.X11R6-SHLIBS
```

```
# X11.X11R6-SHLIBS B.11.00 X11R6 shared libraries
PHSS_15840.X11R6-SHLIBS 1.0 X11.X11R6-SHLIBS applied
PHSS_17422.X11R6-SHLIBS 1.0 X11R6 shared libraries
applied
```

You can list the products and filesets to which a patch applies by listing the `ancestor` attribute. You can generate a list of patches that a given patch superseded by listing the `supersedes` attribute of the patch fileset.

By default, swlist will only show the latest patches installed on a system (i.e., those patches that have not been superseded). To list superseded patches, set the `show_superseded_patches` option to true:

```
swlist -x show_superseded_patches=true
```

Listing Available Patch Categories

You can use the `-l category` specification to list the categories of available patches for patches that are defined with `category` objects.

To list the categories defined for patches in the depot mounted at `/CD`:

```
swlist -d -l category @ /CD
```

```
critical_patch           Patches that fix system
hangsdefect_repair       Provide defect repair
hardware_enablement     Patches enabling new hardware
```

To list a particular attribute of a category object identified by the tag `critical_patch`:

```
swlist -a description -l category critical_patch
```

Using swlist to Resolve Manual Dependencies Some patches cannot automatically resolve their own dependencies. HP-UX patch creation tools mark such patches with a generated tag called `manual_dependencies`. This tag can help you find patches with unresolved dependencies, for example:

```
swlist -l product *,c=manual_dependencies
```

This command lists all patches on your system that need manual resolution of dependencies. To identify the dependencies, list the `readme` attribute for each patch. For example, you could redirect the output of the above command to a file, then use the file to list the readme information for each patch:

```
swlist -l product -a readme *,c=manual_dependencies
```

Patch Removal, Rollback, and Committal

To permit future rollback of a patch, use the `patch_save_files` option. This option (set to true by default) automatically saves any files replaced by a patch. You can then restore the original files if you later decide to remove the patch.

For example,

```
swinstall -s sw_server -x patch_save_files=true
```

Rollback files are saved to the directory:

`/var/adm/sw/save/`*new_patch_name*`/`*fileset_name*

These rules govern patch removal and rollback:

- Using swremove to remove the base fileset of a patch fileset also removes all patches to that fileset.
- Files saved for rollback are also removed when the base fileset to which they apply is updated or removed from the system.
- Removal of a patch automatically rolls back the saved files, unless:
  - You set the `patch_save_files` option to false at the time you installed the patches.
  - You also remove or update the base fileset.
  - You performed a swmodify operation on the patch with the `patch_commit` option set to true.
- You cannot roll back an installed patch that has been superseded unless you first roll back the superseding patch.
- SD performs disk space analysis (DSA) on the save area in the same way it performs DSA on regular file locations.

To save disk space when you are certain a patch operates correctly, you may wish to commit the patch by removing the rollback files saved by the `patch_save_files` option.

To commit a patch, invoke swmodify on the patch with the `-x patch_commit=true` option. (The default value is false.)

For example, to commit the patch `PHKL_1234` and remove its corresponding rollback files:

```
swmodify -x patch_commit=true PHLK_1234
```

NOTE NOTE: When you commit a patch and remove its rollback files, all patches that are superseded by this patch are also committed.

Verifying Patches

The swverify operation on a normal fileset checks that the latest files are properly installed. When installing a patch, the ancestor fileset is updated to have the correct attributes of the patched files.

SD verifies patch filesets by checking that files in a patch are still properly installed (or in the depot correctly).

```
swverify PHCO_1234
```

```
swverify *,c=patch
```

Packaging Patch Software

This section contains information about packaging patch software. Packaging involves the unique patch attributes and behaviors described below.

For complete information on packaging, objects, and attributes, see Chapter 10, “Creating Software Packages,” on page 301.

Patch Software Characteristics

- Each patch fileset only patches files in one base fileset. If a patch needs to modify multiple filesets, the patch product contains a fileset for each base fileset to be modified.
- A patch fileset defines the files to be patched, and the fileset attribute `is_patch` must be set to true.
- The `ancestor` attribute identifies the product or fileset being patched.
- The first patch of any particular patch supersession chain does not have a `supersedes` attribute. A patch that replaces one or more patches has the appropriate `supersedes` attribute.
- All patch software objects with the `is_patch` attribute are automatically assigned the built-in category of `patch`, which is then automatically included in the list of `category_tag` attributes.
- The `category_tag` and `is_patch` attributes at all other levels of software objects besides fileset are for display and selection purposes only. (They are not version-distinguishing attributes.)

Patch Software Objects and Attributes

SD contains attributes specifically for handling patch software. The following attributes are available to all software levels (bundles, products, subproducts, and filesets).

category objects

A software collection can contain a list of category objects, which are used as a selection mechanism. Category objects are identified by the keyword category and contain additional information about this category (a title, tag, and a description of the category). The `category_tag` attribute points to a particular category object and can appear within a product, bundle, subproduct, or fileset.

All software objects with the attribute of `is_patch` set to true are automatically assigned the category of `patch`. (Note that category objects and the `category_tag` attribute can be used independently of patches.)

See “Category Specification” on page 324 for a complete description of category objects.

is_patch

Indicates that a software object is identified as a patch. The default is false. Only filesets with the `is_patch` attribute have patched files. Other levels can be identified as patches for the listing utilities to facilitate identification of patch software at any level.

All software objects with the attribute of `is_patch` set to true are automatically assigned a category of `patch`.

Patch Fileset Attributes

Patch filesets generally operate like normal filesets. Differences are:

- Patch filesets have an explicit ancestor.
- Patch filesets can be installed in the same session as their base, or ancestor, fileset. (The base fileset is always installed first.)
- Patch filesets can be rolled back.
- Patch filesets maintain catalog information to support these features.

- Control scripts delivered with the patch fileset run only when that patch fileset is installed. They do not replace the control scripts for the base fileset.

See “Fileset Specification” on page 331 for a complete description of all patch and non-patch fileset attributes.

Patch fileset attributes include attributes that you can specify in a product specification file (PSF) and attributes generated by SD.

User-specified Attributes

You can specify the following patch fileset attributes in a PSF:

- **ancestor** *`software_specification`*

  Designates the base fileset to be patched.

- **supersedes** *`software_specification`*

  Used when a patch replaces an earlier patch. The attribute indicates which previous patches are replaced by the patch being defined. This attribute is repeatable.

  This attribute consists of a list of software specifications of other patch filesets that the patch supersedes:

`supersedes` ***`product.fileset`*****`,fr=`*****`revision`***

When a patch supersedes another patch, the superseding patch is automatically selected by default. A superseding patch replaces the files of the patch it supersedes when installed after that patch.

Patches may supersede other patches to the same base (non-patch) fileset, or they may be applied to the same base fileset in parallel with other patches.

Patch File Attributes

Patches to the kernel or other libraries can be implemented and removed with the following file level attributes:

type If set to `a`, designates an **archive file** and marks it for an archive action during an install or update. An archive file is a `.o` file that needs to be replaced in an existing archive using the ar command.

archive_path Designates the path to the archive to which the file should be added (instead of installing it to the path location).

When used with the `patch_save_files` option, the `.o` file that previously existed in the archive is saved, and can be restored.

Sample PSF usage:

```
file -t a newfile.o /usr/lib/foolib.a

file
   type a
   archive_path    /usr/lib/foolib.a
   source_path    /usr/lib/newfile.o
```

PSF Example

This sample PSF shows a patch for the file `/build/sbin/who` in the fileset:

```
OS-Core.CMDS-MIN,l=/,r=B.11.00,\
a=HP-UX_B.11.00_32/64,v=HP
```

Note that the HP-UX convention is for patches to use unique product tags but that the fileset tags match those of the ancestor filesets.

```
category

   tag            normal_patch
   revision       0.0
   title          Patches for normal use
   description    Normal patches for typical problems....
end

product
   tag            PHCO_12345
   revision       B.11.00
   architecture   r=B.11.00,a=HP-UX_B.11.00_32/64

   vendor_tag     HP
   title          Core Operating System (patch)
   machine_type   *
   os_name        HP-UX
   os_release     ?.11.*
   os_version     *
   is_patch       true
   category_tag   normal_patch

   fileset
   tag            CMDS-MIN
   revision       B.10.01.001
   title          “Patch of /sbin/who for ... “
   description    “Patch of /sbin/who ... ”
   ancestor       OS-Core.CMDS-MIN,\
                  r=B.10.01_700,a=HP-UX_B.10.01_700,v=HP
   is_patch       true
   file           /build/sbin/who  /sbin/who
   end
end
```

Notes:

- The `ancestor` attribute identifies the fileset to be patched.
- The true value of the `is_patch` attribute at the fileset level flags this fileset as a patch and permits rollback if you use the `patch_save_files` option when you install the patch.

Attributes Generated by SD

SD-UX generates the following patch fileset attributes and stores them in the IPD:

- **applied_patches**

  Set for base (non-patch) software only. Indicates patches that have been applied to a base fileset. An empty list for this attribute indicates the fileset has had no patches applied.

- **applied_to**

  Set for patch filesets only. Indicates the base fileset that the patch was applied to.

- **patch_state**

  Applies to installed patches only. Indicates the current state of the patch:

applied	Patch can be rolled back and has not been superseded.
committed	Rollback files have been deleted.
superseded	Patch has been superseded by another patch.
committed/superseded	Patch has been both committed and superseded.

- **superseded_by**

  Applies to installed patches only. Lists the fileset that caused the current fileset to become superseded.

6 Remote Operations Overview

This chapter presents an overview of remote operations, describing set-up, features, and important concepts to help you effectively manage software across multiple systems. More information about remote operations is also presented in Chapter 7, “Using Jobs and the Job Browser,” on page 215

Table 6-1 Chapter Topics

Topics:
“Introduction” on page 190
“Setting Up Remote Operations” on page 199
“Remote Operations from the Command Line” on page 211
“Using the Remote Operations GUI” on page 193
“Remote Operations Tutorial” on page 200
“Remote Interactive swlist” on page 210

Introduction

In addition to its ability to "pull" software from a central depot, Software Distributor also provides powerful features for remote operations that let you "push" software to remote systems (targets) from the local host. You can use these features interactively and monitor results of all SD-UX commands with the Job Browser or from the command line with the swjob command.

NOTE The Terminal User Interface (TUI) is not available for remote operations.

Differences Between Remote and Local Operations

In general, all Software Distributor features that apply to local operation also apply to remote operations. Additional features of remote operations are summarized in this section.

Remote Targets

For local operations, the target consists of the local host or depots on the local host. For remote operations, the target can be one or more remote systems. A target can also contain depots and act as a source to serve other targets.

Controller, Daemon, and Agent Programs

The controller programs provide the user interface for SD-UX tasks and programs. The controller's role collects and validates data it needs to start a task and to display information on the task's status. The controller also distributes software to remote target machines.

On each target, the SD-UX daemon runs in the background, listening for requests coming from the controller. When a request is received, the daemon schedules the SD-UX agent to perform the task. The daemon also schedules the agent to answer requests from other agent programs that want to use one of the host's depots as a source.

NOTE You must restart the SD-UX daemon if you change daemon options, or the system will not recognize the changes. See “Using Command Options” on page 59 for more information.

Job Management

With SD-UX remote operations, you can create jobs for immediately execution or schedule them for later execution. In addition, you can browse the scheduled, active, and completed jobs using either the command line interface (with the swjob command) or the interactive interface (with the sd command).

Compatible Software

The swconfig, swinstall, and swverify commands let you detect and enforce the use of compatible software (i.e., ensure software products are compatible with system types and operating systems). When you select multiple targets for a remote operation, SD-UX lets you select only the software compatible with *all* targets.

Dependencies Between Software

As with local operations, the swask, swconfig, swcopy, swinstall, swremove, and swverify commands support dependencies between filesets and products. If you have a software selection that specifies a dependency on other filesets or products, the command automatically selects that software. (This step is executed on the local host. You can override this policy with the `autoselect_dependencies` default option.)

With remote operations, dependencies are analyzed on each target and a fileset will not install if dependencies are not met on that target. (You can override this policy using the `enforce_dependencies` default option.)

Session Files

You can use the session file command options to build, save, and reuse sessions with most commands. With remote operations, target selections are saved along with options, source information, and software selections.

Additional GUI Components

SD-UX adds extra components to the GUI programs when remote operations are enabled. Otherwise, the programs are almost identical to those used for local operations. (See “Using the Remote Operations GUI” on page 193.)

Software and Target Lists

Most SD-UX commands let you read lists of software selections from separate input files. With remote operations, you can also read target lists from separate files. The local controller GUI also lets you use software and target lists.

Remote Patch Operations

You can use the SD-UX remote operations features for patch management. You can explicitly select patches for multiple remote systems at one time. Note, however, that the `patch_match_target` option works with only one remote system at a time. See “Installing Patches to Remote Systems” on page 174 for more information.

Limitations

- You cannot use remote operations to directly “push” an HP-UX OS update to remote systems.
- Remote operations do not apply to the following SD-UX commands:
  - install-sd
  - swpackage
  - swmodify

Using the Remote Operations GUI

SD-UX adds extra components to the GUI programs when remote operations are enabled. The extra components for remote operations include a target selection window and features for managing target lists, job preferences, and job monitoring windows. Otherwise, the GUI programs are identical to those used for local operations.

After you set up remote operations and enable the remote operations GUI on the central controller, you can start the swinstall, swcopy, or swremove GUI as you normally would. For example:

```
/usr/sbin/swinstall
```

or

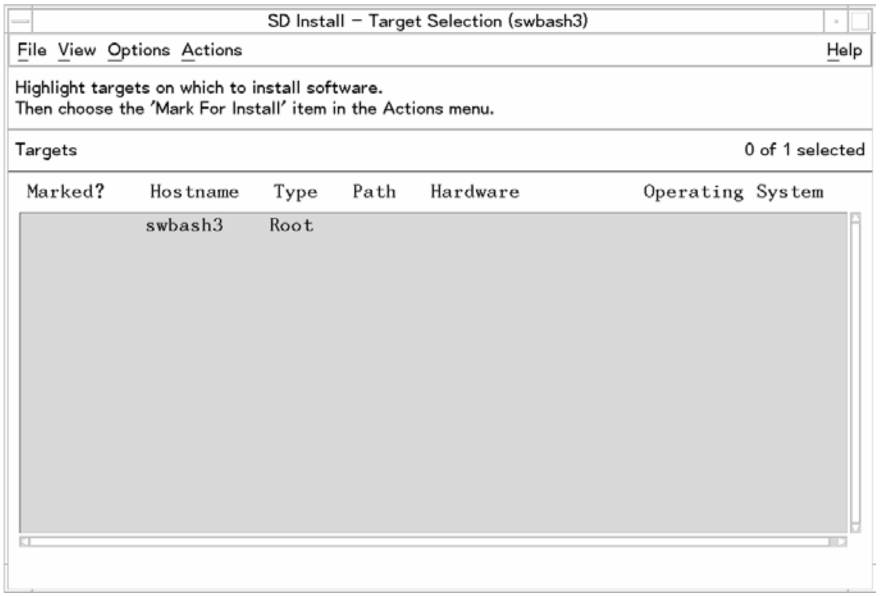
```
/usr/sbin/swinstall -i
```

NOTE The Terminal User Interface (TUI) is not available with remote operations.

Target Selection Window

The Target Selection Window always appears first with the remote GUI programs. Like the Software Selection Window, it features the standard menu bar, message area, and object list of targets available for selection. Instead of selecting software, you select the remote targets on which the remote operation will take place. Menu items and target selection are discussed in the following sections.

Figure 6-1 **Target Selection Window**


SD Install – Target Selection (swbash3)

Highlight targets on which to install software.
Then choose the 'Mark For Install' item in the Actions menu.

Targets 0 of 1 selected

```
Marked?   Hostname   Type   Path   Hardware   Operating System
          swbash3    Root
```

Performing Actions

The general procedure for using the remote operations GUI is to:

1. Select one target at a time by highlighting a target in the object list of the Target Selection Window.
2. Select **Actions→Mark for Install....** (or **Actions→Mark for Copy....** or **Actions→Mark for Remove....**).
3. Repeat 1 and 2 for any additional targets.

4. When you have selected all targets for your operation, select **Actions→Show Software for Selection....** to display the Software Selection Window.

Selecting Multiple Targets

This section discusses how to install to multiple targets, create target groups, and how to save these groups for future software installations. (For single-target installations to your local (default) target, see the procedures in "Remote Operations Tutorial" on page 200).

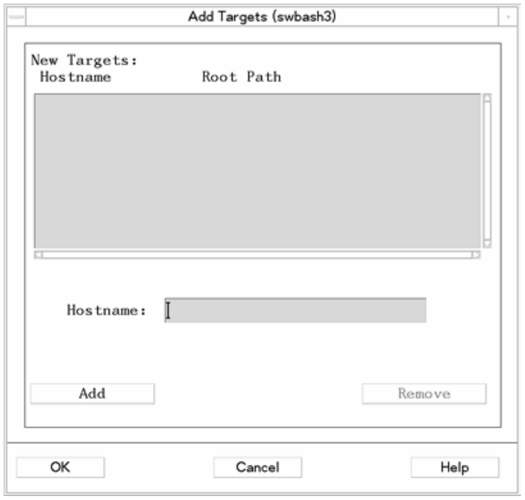
The Target Selection Window displays a list of targets may be displayed:

- If you have recalled a session file (**File→Recall Session**), any hosts defined in that session are displayed.
- Otherwise, any hosts specified in the default hosts file (`/var/adm/sw/defaults.hosts` or `$HOME/.sw/defaults.hosts`) are displayed. (See "Preselecting Host Files" on page 39.)
- If you started SD-UX from ServiceControl Manager, targets are pre-selected and cannot be changed.

If the desired target for the installation is not in the list:

1. Choose **Actions→Add Targets....** The Add Targets dialog (Figure 6-2, "Add Targets Dialog,") is displayed.

Figure 6-2 Add Targets Dialog



2. Enter the primary root name in the `Hostname:` area and select **Add.**
3. The Select Target Path dialog appears. The default path is root (`/`). To accept the default root (`/`), click **OK**.
4. After selecting the root path, the Hostname and Root Path are automatically updated in the Add Target dialog (Figure 6-2, "Add Targets Dialog,"). To add additional targets, repeat 2.
5. Select **OK** in the Add Targets dialog. This adds your selections to the Target Selection Window. Each target is contacted as it is added to the Target Selection Window. Networking may cause delays; if the SD-UX daemon is not running on the target, the delay lasts until the daemon times out.

   From the Target Selection Window, any targets added using **Add Targets...** are automatically marked `Yes`.
6. If there are any other desired targets in the Target Selection List that are not marked and you want to install to them, highlight the target by clicking on it.

   Choose **Actions**→**Mark for Install**. The `Marked?` column is set to `Yes` for that target.

   — or —

   Hold down the *right* mouse button and choose **Mark for Install** from the resulting menu.
7. To unmark a target in the Target Selection Window (i.e., object list):
   a. Highlight the target
   b. Choose **Actions**→**Unmark for Install**. The `Marked?` column is cleared for that target.

      — or —

      Hold down the *right* mouse button and choose **Unmark for install**.

At this point, all desired targets should be listed and have `Yes` in the `Marked?` column. If you have not marked any targets, you cannot proceed to the Selecting Software phase.

Selecting Individual Targets

You can add or delete individual targets.

To add a new target:

1. Select **Actions**→**Add Targets...**.The Add Targets dialog appears.
2. Type in the name of the desired target and click on **Add**. The Select Target Path dialog appears.
3. Click **OK** to accept the default (/) or click on the **Root Path...**button to display the Shared Root Paths dialog, which contains more selection options.
4. Select the desired root and click **OK** to return to the Select Target Path dialog.
5. Click **OK** to return to the Add Targets dialog
6. Click **OK**. You have now marked an additional target.

To delete targets, select one or more targets from the Target Selection Window, then select **Actions**→**Unmark for Install**.

Saving a Target Group

You may want to re-use your list of targets for a later session. To do so,

1. Select **Actions→Save Target Group...**

   The Select File dialog appears. If target groups already exist, the first file path appears in the text box in the bottom of the dialog. Type a name for a new group or re-use an existing group (saving your current list to existing target group overwrites that group). Groups are saved in the directory:

   `$HOME/.sw/targets`

2. To save the group, click **OK**.

   This saves all the target selections you have just marked (all targets listed with `Yes` in the `Marked?` Column). This group will automatically appear in the Select File dialog for all subsequent target group selections.

Adding a Target Group

To re-use a target group that you previously saved:

1. Select **Actions→Add Target Group...**. The Select File dialog appears. All existing target groups appear in the list.
2. Select the target group you want and click **OK.**

   The targets from that group are now marked, along with any other targets you had already marked.

Setting Up Remote Operations

SD-UX uses Access Control Lists to authorize anyone who is attempting to create, modify, or read software products in a depot or to install software to a root file system. (ACLs are discussed in detail in Chapter 9, “SD-UX Security,” on page 255.) To enable the remote operations, you must install a special HP ServiceControl Manager fileset on each remote system to be managed. You can then enable the remote operations GUI.

1. As root, enter the following command on the controller system:

   ```
   swreg -l depot /var/opt/mx/depot11
   ```

   - If the remote host is running HP-UX 10.20, use the same command but substitute **`depot10`** for **`depot11`**.
   - This sets up sharing of the depot used to enable the remote systems.

2. As root, enter the following command on each remote system to be managed. This sets up the root, host and template ACLs in a way that permits root access from the controller system:

   ```
   swinstall -s central_node:/var/opt/mx/depot11 \
   AgentConfig.SD-CONFIG
   ```

   - In this example, `central_node` is the name of the controller.
   - If the remote host is running HP-UX 10.20, use the same command but substitute **`depot10`** for **`depot11`**.
   - Remote systems previously set up with OpenView Software Distributor do not require this step.
   - Software Distributor does not require any other ServiceControl Manager filesets.

3. (Optional) On the central controller system only, enter the following command to enable the remote operations GUI interface:

   ```
   touch /var/adm/sw/.sdkey
   ```

   (This step is not required when you use SD-UX from within the HP ServiceControl Manager.)

Remote Operations Tutorial

This tutorial introduces you to the remote operations user interface and to the general flow for distributing software to other systems. Also, you will learn how to preview, schedule, and monitor your distribution jobs. Although this tutorial uses swinstall for the example GUI, the swcopy and swremove GUI programs are almost identical. You can apply the knowledge you gain from this tutorial to those tasks.

You may wish to go through this tutorial more than once to experiment with variations in the basic operations.

Tutorial Set-Up

1. Set up remote operations on your controller system and a remote test system. (See “Setting Up Remote Operations” on page 199.)
2. As root on the controller system, enable the remote operations GUI:

   ```
   touch /var/adm/sw/.sdkey
   ```

3. Make sure your PATH variable contains `/usr/sbin`. To check, enter:

   ```
   echo $PATH
   ```

4. Make sure your DISPLAY variable is properly set by typing:

   ```
   echo $DISPLAY
   ```

5. Ensure that the examples are installed. Enter:

   ```
   swlist SW-DIST.SD-EXAMPLES
   ```

6. Create the depot containing example package (i.e., SD-DATABASE):

   ```
   cd /usr/lib/sw/examples/swpackage/depot_src
   swpackage -s psf @ /var/adm/sw/examples/depot
   swreg -l depot @ /var/adm/sw/examples/depot
   ```

7. To verify that the software is in the depot and is available for distribution to targets, enter:

   ```
   swlist -s /var/adm/sw/examples/depot
   ```

   You should see `SD-DATABASE` in the resulting list.

How to Perform a Single-Target Installation

Overview

The tutorial consists of these steps:

Table 6-2 **Installation Steps**

Overview of Installation Steps	
I. Start-up	Start the Job Browser.
II. Select Targets	Specify the targets where you want the software installed. You can use the default local target or specify another target.
III. Select Source	Provide the location of the software depot from which the software will be installed with the Specify Source dialog.
IV. Select Software	Use the Software Selection Window to select the software to install.
V. Specify Install Preferences	Use the Install Preferences dialog box to set preview or scheduling options.
VI. Analysis and Installation	Perform the actual software installation or preview.
VII. Monitor Results	Monitor job progress and results using the Job Browser GUI.
VIII. Remove Jobs	Delete the completed jobs using the Job Browser.

Step I: Start-up

To initiate an install session:

1. Start the Job Browser by typing:

   sd

2. From the Job Browser window, choose

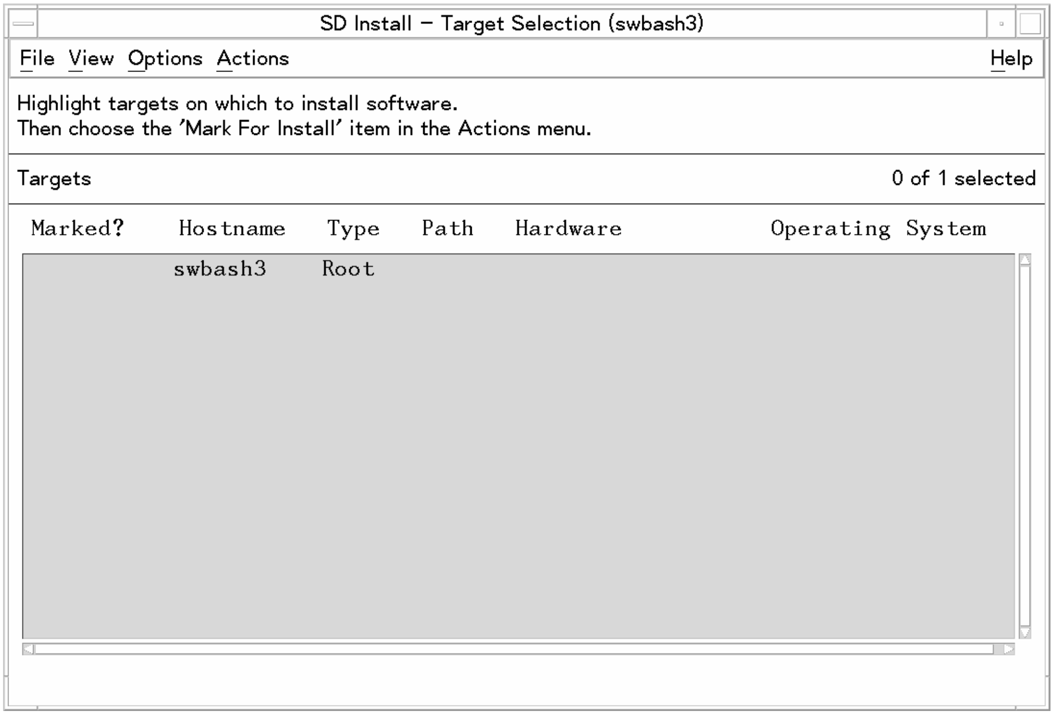
   Actions→Create Job→Install Software...

   The message `Invoking a swinstall process` displays at the bottom of the window, then the Target Selection Window appears.

Step II: Select Targets

The Target Selection Window displays the local, default target. A target is where you want the installation to go (in the example below, the target is the system swbash3). By default, the current system is listed (Figure 6-3, “Target Selection Window,”).

Figure 6-3 **Target Selection Window**


SD Install – Target Selection (swbash3)

Highlight targets on which to install software.
Then choose the 'Mark For Install' item in the Actions menu.

Targets 0 of 1 selected

```
Marked?    Hostname   Type   Path   Hardware          Operating System
           swbash3    Root
```

Specify the desired target for the installation:

1. **For local default**:

   a. Highlight the local target system with a left mouse click. Then select **Actions→Mark for Install** (or right-click to display the pop-up menu and select **Mark for Install**).

   b. Select **Actions→Show Software for Selection...**

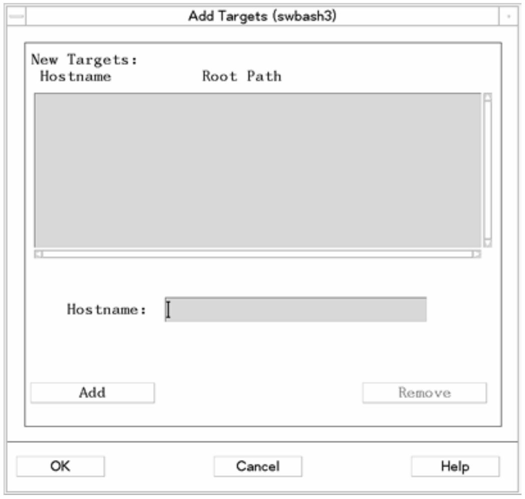
   This displays the Specify Source dialog. If this is your first time through this tutorial, skip directly to “Step III: Select Source” on page 204. After you have gone through this tutorial once, retry Step I using remote targets.

— *or* —

For remote targets: choose **Actions→Add Targets** to install to a different target. This takes you to the Add Targets dialog (Figure 6-4, “Add Target Dialog (for multiple or non default targets),”).

2. Enter the target name in the `Hostname:` area (e.g., `system_two`) and select **Add**. This takes you to the Select Target Path dialog.

Figure 6-4 Add Target Dialog (for multiple or non default targets)

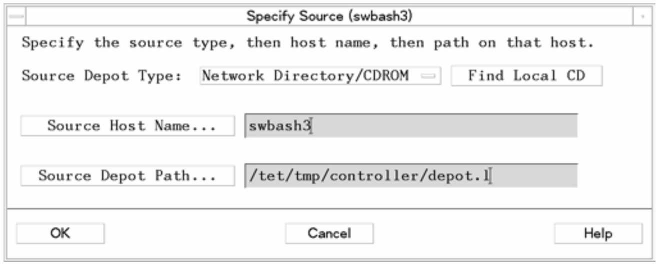


3. Use the current root path (/) by selecting **OK**. This returns you to the Add Targets dialog.
4. Select **OK** in the Add Targets dialog. This updates the Target Selection Window with your target selection. `Yes` appears in the Marked column, indicating that the target is marked for installation.
5. Choose **Actions→Show Software for Selection**. The Specify Source dialog appears.

Step III: Select Source

In this step, the Specify Source dialog lets you select the Source Host Name (the source system where the depot resides) and Source Depot Path (path of the depot containing the software).

Figure 6-5 Specify Source Dialog



Specify Source (swbash3)

Specify the source type, then host name, then path on that host.

Source Depot Type: Network Directory/CDROM

Source Host Name...: swbash3

Source Depot Path...: /tet/tmp/controller/depot.1

1. The Specify Source dialog should list your controller name or your remote test system name in the **Source Host Name...** field and the example depot that you created (`/var/adm/sw/examples/depot`) in the **Source Depot Path...** field.

   From this dialog, you can also:

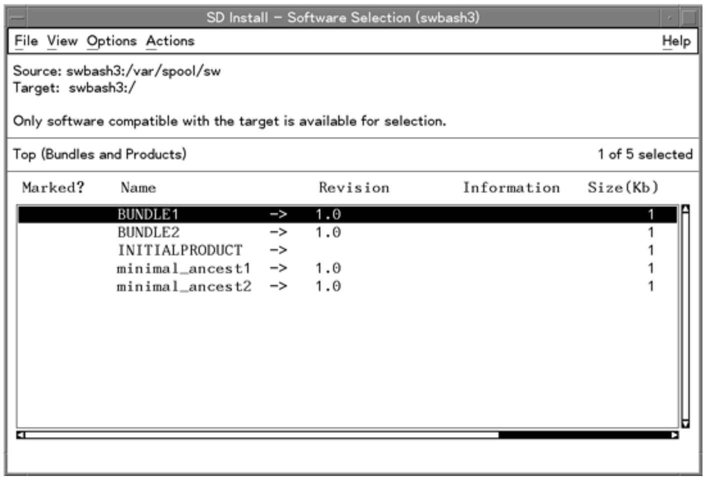
   - Click on the **Source Host Name...** button to display a list of hosts that you can select from.
   - Click on the **Source Depot Path...** to display a list of registered depots that you can select from.

   Click **OK**. The Software Selection Window appears (Figure 6-6, “Software Selection Window,”). This window displays all available software in the depot that you selected.

Step IV: Select Software

Use the Software Selection Window to select the software you want to install.

Figure 6-6 **Software Selection Window**



SD Install – Software Selection (swbash3)

File View Options Actions Help

```
Source: swbash3:/var/spool/sw
Target:  swbash3:/

Only software compatible with the target is available for selection.

Top (Bundles and Products)                                  1 of 5 selected

Marked?   Name                     Revision     Information   Size(Kb)
          BUNDLE1            ->    1.0                               1
          BUNDLE2            ->    1.0                               1
          INITIALPRODUCT     ->                                      1
          minimal_ancest1    ->    1.0                               1
          minimal_ancest2    ->    1.0                               1
```

1. Highlight `SD-DATABASE` (i.e., the example software) by clicking on it with the left mouse button.
2. Choose **Actions→Mark for Install** (or right-click to display the pop-up menu and select **Mark for Install**).

   The `Marked?` column is set to `Yes` for `SD-DATABASE`.

Table 6-3 **Software Selection List**

Software Selection Window Object List
The Software Selection Window object list is hierarchical: you can open each object in the list and show objects contained inside. Objects in the list that contain other objects that can be opened, have an arrow (→) after the name.

Table 6-3 **Software Selection List (Continued)**

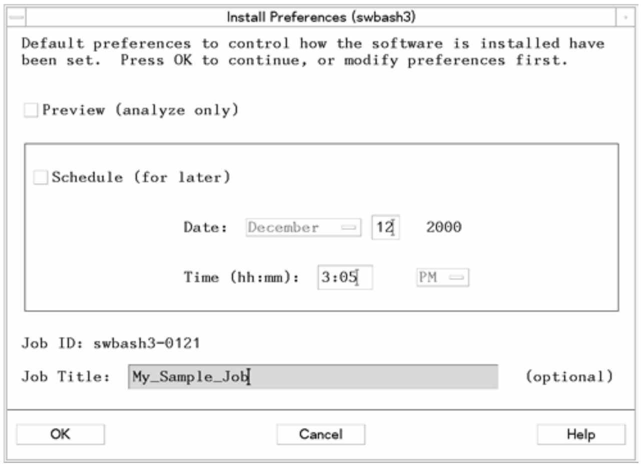
Software Selection Window Object List
For example: • To see the subproducts in the product `SD-DATABASE`, double click on it. The object list displays the subproducts. To open a subproduct, double click on the name. (Or highlight the name and then select **Actions→Open Item**...) • To close an object and return to the previous list, double click on the first item in the list (`..(go up)`) or highlight the item and selecting **Actions→Close Level**. Note that products are listed together, but subproducts and filesets may appear in the same list when you open a product.

3. Choose **Actions→Install**. This displays the Install Preferences dialog (Figure 6-7, "Install Preferences Dialog,").

Step V: Specify Install Preferences

The Install Preferences dialog box gives you the following optional selections: **Preview**, **Schedule**, and **OK.** You can also enter a Job Title.

Figure 6-7 **Install Preferences Dialog**


Install Preferences (swbash3)

Default preferences to control how the software is installed have been set. Press OK to continue, or modify preferences first.

☐ Preview (analyze only)

☐ Schedule (for later)

Date: December 12 2000

Time (hh:mm): 3:05 PM

Job ID: swbash3-0121

Job Title: My_Sample_Job (optional)

1. Select the text area after **Job Title** and type:

   SDTESTJOB

This is the name of your install job.

2. Select **OK** to install the software now.

   For single-target installations such as this tutorial, the Install Analysis dialog appears (Figure 6-8 on page 208).

3. If this is your first pass through the tutorial, proceed to Step V.

4. (Optional) Previewing a Job

   a. Select the **Preview** button. This tells SD to analyze the software without installing it.

   b. Click **OK**. The Install Analysis dialog appears. This dialog lets you monitor the analysis of a single-target job. You can also browse log files and product summary information.

   c. When the target Status indicates Ready (analysis is successful), select **OK**. This returns you to the Software Selection Window.

   d. Select **Actions→Install**. The Install Preferences dialog appears.

   e. Proceed with the installation by selecting **OK** in the Install Preferences dialog.

      (If you repeat this tutorial and choose to preview a job that uses multiple targets, you will find that the Install Analysis dialog does not appear. You can only monitor the preview job progress from the Job Browser. See “Step VII: Monitor Results” on page 209 for more information.)

5. (Optional) Scheduling a Job

   a. Select the **Schedule** button. This activates the fields that let you specify the time and date you at which you want your job to run. (For example, you may want to schedule a job at midnight when few users are logged in.)

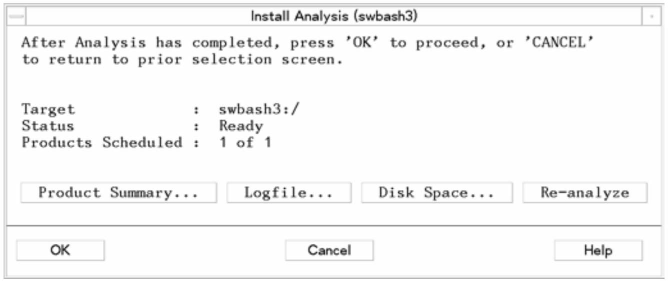
   b. After you specify the schedule information, click **OK**. The system displays a note indicating that the job has been scheduled.

   c. Click **OK** in the dialog. The Target Selection Window reappears. Select **File→Exit** to return to the Job Browser, from which you can monitor your scheduled job.

Step VI: Analysis and Installation

SD-UX analyzes the target before performing the actual install, copy, or remove operation. (If you set up a preview job in Step IV, the install stops after the analysis.)

Figure 6-8 **Install Analysis Dialog**



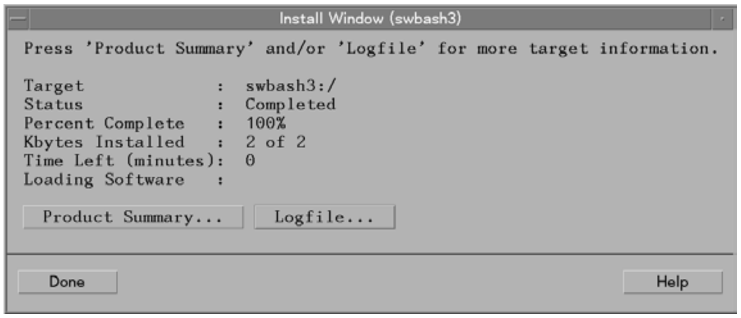
```
After Analysis has completed, press 'OK' to proceed, or 'CANCEL'
to return to prior selection screen.

Target             :  swbash3:/
Status             :  Ready
Products Scheduled :  1 of 1
```

1. When the Analysis is complete, the status for the target you selected should show `Ready`, indicating no errors or warnings occurred during analysis. Select **OK** to proceed with the installation.

   The Install Window dialog (Figure 6-9, “Install Window dialog,”) appears, and the installation starts automatically. When the status in the dialog changes to `Completed`, the installation has successfully completed.

Figure 6-9 **Install Window dialog**



```
Press 'Product Summary' and/or 'Logfile' for more target information.

Target               :  swbash3:/
Status               :  Completed
Percent Complete     :  100%
Kbytes Installed     :  2 of 2
Time Left (minutes):  0
Loading Software     :
```

2. Select **Done** to exit the Install Window dialog. This returns you to the Target Selection Window.
3. Select **File**→ **Exit** to return to the Job Browser.
4. (Optional) Select another target for installation (i.e., **Actions**→**Mark for Install**).

Step VII: Monitor Results

When you exit the Target Selection Window, you return to the Job Browser. The icons in the job list change to show the status of jobs. Different icons indicate different job status. (See “Job Browser Icons” on page 218 for sample icons.)

Your job, labeled `SDTESTJOB`, should show with either a check mark or a ruler icon. To verify status information for `SDTESTJOB` from the job list:

- Double click `SDTESTJOB` to invoke the Job Results dialog.
- Double click the target to show the detailed target log.
- Click **OK** to close each dialog after you have viewed it.

— *or* —

1. Select the `SDTESTJOB` icon.
2. Choose **Actions→Show Job Description**.... The Job Description dialog appears. This displays all of the job attributes, the software and the target(s) involved.
3. (Optional) Select **Show Options...** to see what the job option settings.
4. (Optional) Select **Show Results...** to see the latest job status.

Step VIII: Remove Jobs

After you have run the tutorial, use the Job Browser to remove the example jobs:

1. Click on the `SDTESTJOB` icon.
2. Select **Actions→Remove Job...**. The Remove a Job dialog box appears.
3. Select **OK**.

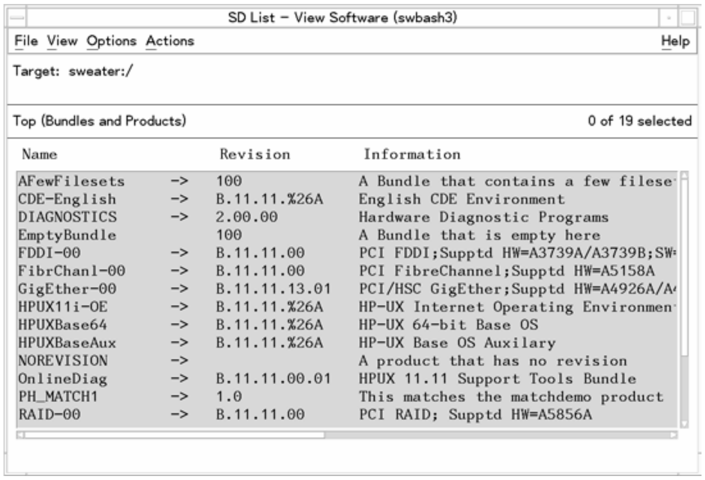
— *or* —

1. Select the job icon and right click.
2. Select **Actions→Remove Job...** from the pop-up menu. The Remove a Job dialog appears, displaying `SDTESTJOB`.
3. Select **OK**. The `SDTESTJOB` icon disappears from the Job Browser and the job is removed from the SD-UX database.

Remote Interactive swlist

For remote operations, the `swlist -i` command starts a list browser that lets you interactively list installed software on remote hosts. The only difference between remote and local operations is the name of the target displayed in the message area of the Software Browsing window.

Figure 6-10 **The swlist Browser**



```
Target: sweater:/

Top (Bundles and Products)                                 0 of 19 selected

Name                  Revision         Information
AFewFilesets     ->   100              A Bundle that contains a few filese
CDE-English      ->   B.11.11.%26A     English CDE Environment
DIAGNOSTICS      ->   2.00.00          Hardware Diagnostic Programs
EmptyBundle           100              A Bundle that is empty here
FDDI-00          ->   B.11.11.00       PCI FDDI;Supptd HW=A3739A/A3739B;SW
FibrChanl-00     ->   B.11.11.00       PCI FibreChannel;Supptd HW=A5158A
GigEther-00      ->   B.11.11.13.01    PCI/HSC GigEther;Supptd HW=A4926A/A
HPUX11i-OE       ->   B.11.11.%26A     HP-UX Internet Operating Environmen
HPUXBase64       ->   B.11.11.%26A     HP-UX 64-bit Base OS
HPUXBaseAux      ->   B.11.11.%26A     HP-UX Base OS Auxilary
NOREVISION       ->                    A product that has no revision
OnlineDiag       ->   B.11.11.00.01    HPUX 11.11 Support Tools Bundle
PH_MATCH1        ->   1.0              This matches the matchdemo product
RAID-00          ->   B.11.11.00       PCI RAID; Supptd HW=A5856A
```

For more information about swlist, see “Listing Your Software (swlist)” on page 96.

Remote Operations from the Command Line

Running remote operations from the command line is almost identical to those for local operations. Key differences are:

- You must specify target selections.
- You can monitor jobs using the swjob command, as discussed in "Monitoring Jobs from the Command Line" on page 230.
- You can use additional command options to schedule and manage jobs. See "Managing and Tuning Jobs with Command Options" on page 234.

Target Selections

By definition, you must specify a remote target for a remote operation. Unlike local operations, in which a target could be a directory on the local host system, you must specify remote systems as targets for remote operations.

```
swinstall -s sw_server cc pascal @ hostA hostB hostC
```

(This installs the C and Pascal products onto three remote hosts.)

Syntax

Software and source depot selections are followed by target selections. These operands are separated by the "@" (at) character. This syntax implies that the command operates on selections at targets.

The *target_selections* syntax is identical for all Software Distributor commands that require it:

`@ [`*`host`*`][:][/`*`directory`*`]`

- Only one @ character is needed.
- You can specify the host by its host name, domain name, or internet address. A directory must be specified by an absolute path.
- The `:` (colon) is required if you specify both a host and directory.

- On some systems, the @ character is used as the kill function. Type `stty` on your system to see if the @ character is mapped to any other function on your system. If it is, remove the mapping, change the mapping, or use `\@`.

Target Files

You can also use an input file to specify targets. To keep the command line shorter, target selection input files let you specify long lists of targets. With a target selection file, you only have to specify the single file name.

The `-t` command-line option lets you specify a target file. For example:

```
swinstall -f mysoft -s /mnt/cd -t mytargs
```

In this example, the file `mytargs` (which resides in the default directory) contains a list of target selections for the depot /mnt/cd.

In the target file, blank lines and comments (lines beginning with #) are ignored. Each target selection must be specified on a separate line and must consist of a host name or network address, optionally followed by a colon and a full path:

host[:*/directory*]

Examples

swacl

To list the global product template ACL on remote host `gemini`:

```
swacl -l global_product_template @ gemini
```

swask

To run all request scripts from depot `/var/spool/sw` on the remote system `swposix` and write a response file back to the same depot:

```
swask -s swposix:/var/spool/sw \*
```

swconfig

To configure the C and Pascal products on three remote hosts:

```
swconfig cc pascal @ hostA hostB hostC
```

swcopy

To copy the C and Pascal products to one local and two remote depots:

```
swcopy -s sw_server cc pascal @ /var/spool/sw \
       hostA:/tmp/sw hostB
```

swinstall

To install the C and Pascal products to three remote hosts:

```
swinstall -s sw_server cc pascal @ hostA hostB hostC
```

swjob

The swjob command lets you monitors jobs from the command line. For more information about jobs, see Chapter 7, "Using Jobs and the Job Browser," on page 215 and "Monitoring Jobs from the Command Line" on page 230.

To list the agent log of remote system `TargetA` for job `hostA-0001`:

```
swjob -a log hostA-0001 @ targetA:/
```

swlist

To list the C product on three remote hosts:

```
swlist cc @ hostA hostB hostC
```

swreg

To unregister the default depots on three remote hosts:

```
swreg -u -l depot /var/spool/sw @ hostA hostB hostC
```

swremove

To remove the C and Pascal products from three remote hosts:

```
swremove cc pascal @ hostA hostB hostC
```

swverify

To verify the C and Pascal products on three remote hosts:

```
swverify cc pascal @ hostA hostB hostC
```

7 Using Jobs and the Job Browser

This chapter describes SD-UX jobs the Job Browser interface for remote operations. For additional information on remote operations, see Chapter 6, “Remote Operations Overview,” on page 189.

Table 7-1 Chapter Topics

Topics:
“Introduction” on page 216
“Using the Job Browser” on page 217
“Monitoring Jobs from the Command Line” on page 230
“Managing and Tuning Jobs with Command Options” on page 234

Introduction

The Job Browser GUI, an interactive interface for managing remote operations. The Job Browser lets you:

- Create copy, install, or remove jobs
- Monitor job status and logfiles
- List job information
- Schedule jobs

The swjob command lets you monitors jobs from the command line. Various command options help you manage and tune performance of jobs and remote operations.

NOTE The Terminal User Interface (TUI) has some limitations when used with the Job Browser:

- Error-handling messages may garble the screen. Type **Ctrl-L** to refresh the screen if this happens.
- If you display **Actions**→**Job Description**→**Show Options**, some scrolling is required to view the entire screen.

Starting the Job Browser

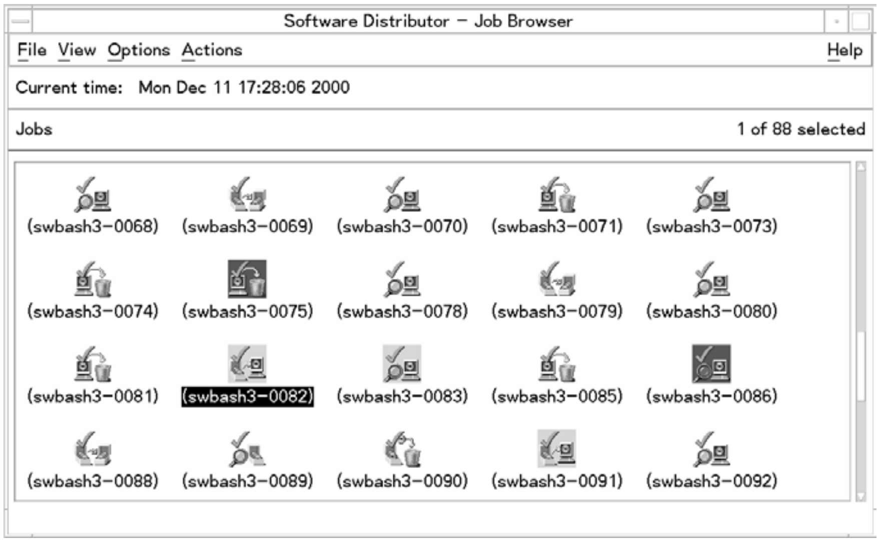
To start the Job Browser, type:

```
sd
```

The Job Browser window displays on your screen.

Using the Job Browser

Figure 7-1 The SD Job Browser Window


Software Distributor – Job Browser
File View Options Actions Help
Current time: Mon Dec 11 17:28:06 2000
Jobs 1 of 88 selected
(swbash3–0068) (swbash3–0069) (swbash3–0070) (swbash3–0071) (swbash3–0073)
(swbash3–0074) (swbash3–0075) (swbash3–0078) (swbash3–0079) (swbash3–0080)
(swbash3–0081) (swbash3–0082) (swbash3–0083) (swbash3–0085) (swbash3–0086)
(swbash3–0088) (swbash3–0089) (swbash3–0090) (swbash3–0091) (swbash3–0092)

The window is divided into three parts:

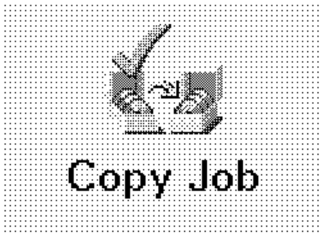
- Menu bar, which contains most of the standard SD-UX menus discussed in “Using the GUI and TUI Commands” on page 35. Menu items specific to the Job Browser are discussed later in this chapter.
- Message area, which displays the current time.
- Jobs List, which displays job icons (the default) representing each job. Under the icon is the job title. To select a job, click on the icon. See “Job Browser Icons” on page 218.

NOTE Until a job is created, the Jobs List is empty.

Job Browser Icons

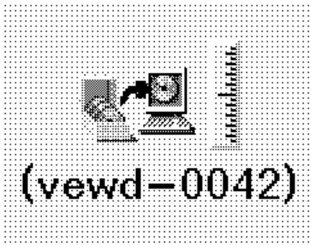
- A clock indicates that the job is scheduled but hasn't run yet.
- A check mark indicates that a job has completed.
- A ruler indicates that the job is active.
- A red background indicates that the job contained errors.
- A yellow background indicates that the job contained warnings.

Figure 7-2 **Copy Icon**



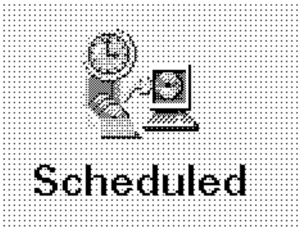
This icon represents a copy job (depot to depot). A check mark indicates that the job has completed.

Figure 7-3 **Active Install Job Icon**



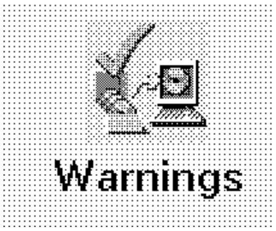
This icon represents an install job. The ruler on the side indicates that the job is active.

Figure 7-4 **Scheduled Install Job Icon**



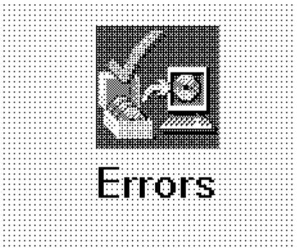
This icon represents an install job that is scheduled for a later time. The clock face indicates that it is a scheduled job.

Figure 7-5 **Install Job with Warnings Icon**



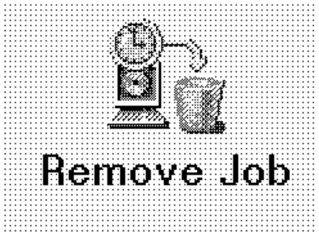
This icon represents an install job that completed, but contained warnings. The background around the icon is yellow.

Figure 7-6 **Install Job with Errors Icon**



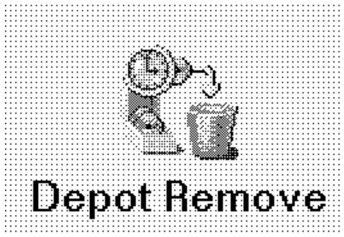
This icon represents an install job that completed, but contained errors. The background around the icon is red.

Figure 7-7 **Scheduled Remove Installed Software Job Icon**



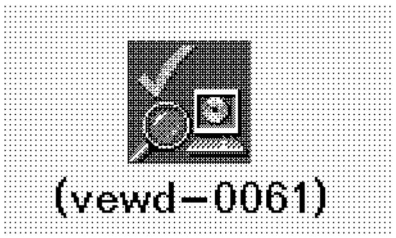
This icon represents a scheduled remove job on installed software.

Figure 7-8 **Scheduled Remove Depot Software Job Icon**



This icon represents a scheduled remove job on software contained in a depot.

Figure 7-9 **Verify Job Icon**



This icon represents a verify job (represented by a magnifying glass) that completed, but contained errors. The background around the icon is red.

The File Menu

The **File** menu has the following options:

Search	Performs text searches for job IDs or titles.
Print	Lets you print the jobs list.
Exit	Exits the Job Browser

Printing the Jobs List

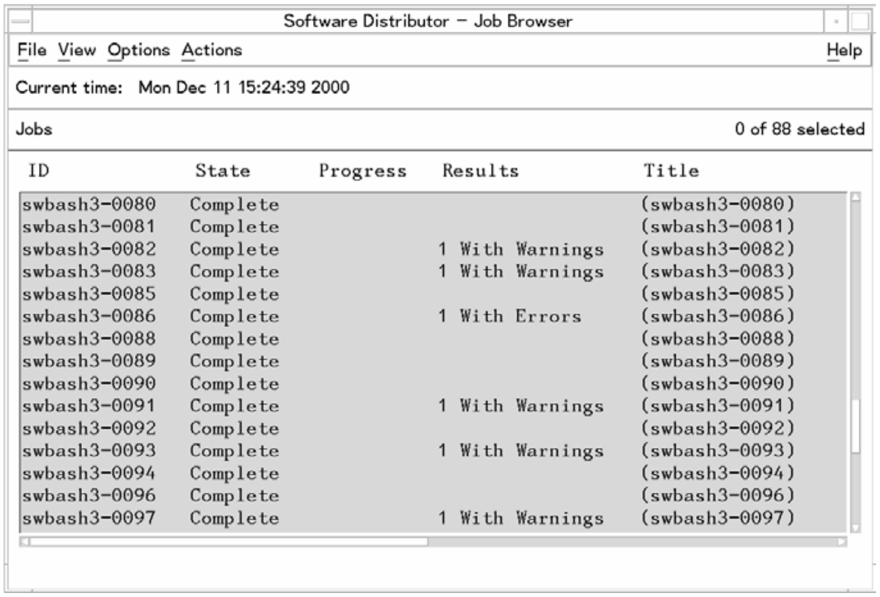
This option prints the Jobs List to a specified printer or saves it to a file. (The Jobs List can only be printed if it is listed by properties—see “The View Menu” on page 222.) If the Jobs List is displayed by name and icon (the default), this menu item is greyed-out and cannot be chosen. To print the Jobs List:

1. Select **View**→**By Properties**.
2. Choose **File**→**Print**.... The Print Objects dialog is displayed.
3. Supply any necessary information in the Print Objects dialog and select **OK**.

The View Menu

The View menu lets you change the way information is presented in the Job Browser. The standard choices on this menu (**Columns...**, **Filter...** , **Sort...** and **Save View as Default**) match those described in “Changing Software Views—The View Menu” on page 42. Note, however, that the **Columns...** choice is only valid for **View→By Properties** (discussed below).

Figure 7-10 Jobs Displayed by Properties



Current time: Mon Dec 11 15:24:39 2000

Jobs 0 of 88 selected

```
ID             State      Progress   Results           Title
swbash3-0080   Complete                                (swbash3-0080)
swbash3-0081   Complete                                (swbash3-0081)
swbash3-0082   Complete              1 With Warnings   (swbash3-0082)
swbash3-0083   Complete              1 With Warnings   (swbash3-0083)
swbash3-0085   Complete                                (swbash3-0085)
swbash3-0086   Complete              1 With Errors     (swbash3-0086)
swbash3-0088   Complete                                (swbash3-0088)
swbash3-0089   Complete                                (swbash3-0089)
swbash3-0090   Complete                                (swbash3-0090)
swbash3-0091   Complete              1 With Warnings   (swbash3-0091)
swbash3-0092   Complete                                (swbash3-0092)
swbash3-0093   Complete              1 With Warnings   (swbash3-0093)
swbash3-0094   Complete                                (swbash3-0094)
swbash3-0096   Complete                                (swbash3-0096)
swbash3-0097   Complete              1 With Warnings   (swbash3-0097)
```

Viewing By Name and Icon

- Choosing **View→By Name and Icon** displays the jobs list in name and icon format.
- This menu item and the **View→By Properties** menu choice operate as radio buttons. When one is chosen, the other is un-chosen.

Viewing By Properties

- Choosing **View→By Properties** displays the jobs list by properties (job title, ID, type of operation, scheduled date, status, progress, results, and date of last update).

- Any modifications made in the **View→Columns...**, **View→Filter**..., or **View→Sort**... menu selections affect how the property list is displayed.
- This menu item and the **By Name and Icon** menu choice operate as radio buttons. When one is chosen, the other is un-chosen.
- You can print the job list from this view by choosing **File→Print**....

The Options Menu

The **Options** menu optional behavior of the Job Browser.

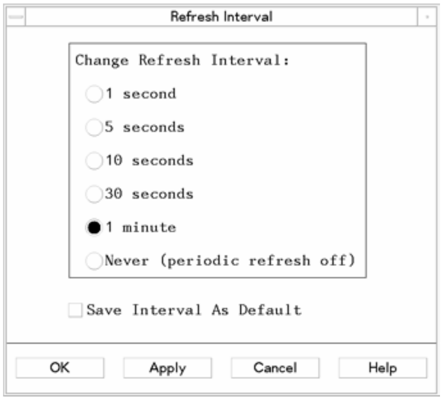
Changing the Refresh Interval

By default, the Jobs List is refreshed every minute. You may want the list updated more frequently if you are monitoring a lot of jobs. Or, you can turn off the automatic refresh feature to improve performance.

To change how often the list is updated:

1. Choose **Options→Change Refresh Interval**.... The Refresh Interval dialog is displayed.
2. Select a new refresh interval from the list.

Figure 7-11 **Refresh Interval Dialog**



- **Apply** immediately applies the interval you have selected.
- **Save Interval as Default** sets the selected refresh interval as the default for future sessions.

To change the refresh interval for the SD-UX daemon, see “Managing and Tuning Jobs with Command Options” on page 234.

Refreshing the Jobs List

To immediately update the Jobs List, choose **Options**→**Refresh List**.

The Actions Menu

Items in the **Actions** menu let you perform job creation and management tasks. If you have selected a job selected, the actions available apply specifically to that job. If you do not have a job selected, the only action available is job creation.

Shortcuts

To display a pop-up menu of job-specific actions, right-click on a job icon, then left-click. This displays a pop-up **Actions** menu items. Choose an action by clicking with either mouse button.

Double clicking on a job displays the Job Results dialog (same as **Actions**→**Show Job Results**....)

Creating a Job

To create a job, choose **Actions**→**Create Job**. This brings up a submenu with the following choices that start different sessions:

Table 7-2 **Job Actions Options**

Job Actions & Selections	
Install Software...	swinstall session.
Remove Installed Software...	swremove session.
Copy Software to a Depot...	swcopy session.
Remove Software from a Depot...	swremove -d session.

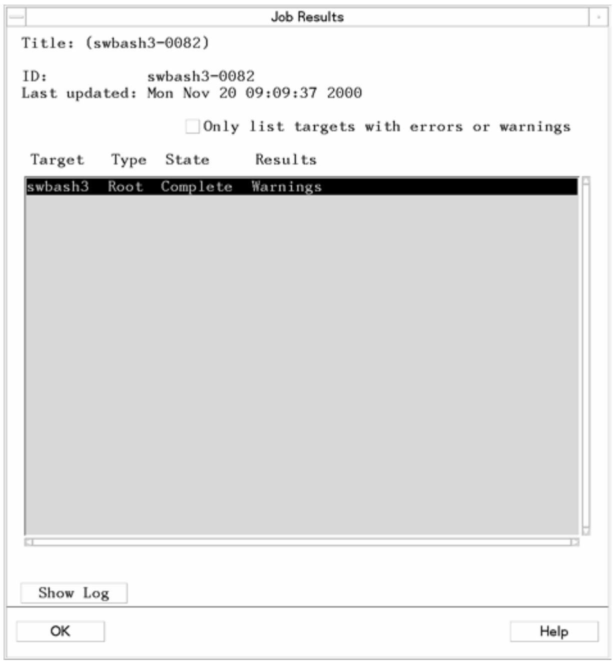
Showing Job Results

Selecting **Actions**→**Show Job Results...** displays the Job Results dialog, which lists results for the job selected. (You can also reach this dialog by double-clicking a Job Browser icon.)

- The object list shows the list of targets for the job, their type, and job status.
- The **Show only warnings or errors** toggle button changes the information displayed to show all targets or to show only the list of targets with warnings or errors in the job.

- Select the **Show Log** button or double-click on a target in the object list opens the log for that target.
- Select **OK** to return to the Job Browser. (Closing the dialog does not stop the jobs displayed if they are active.)

Figure 7-12 Job Results Dialog



```
Title: (swbash3-0082)

ID:           swbash3-0082
Last updated: Mon Nov 20 09:09:37 2000

 Target    Type  State     Results
swbash3   Root  Complete  Warnings
```

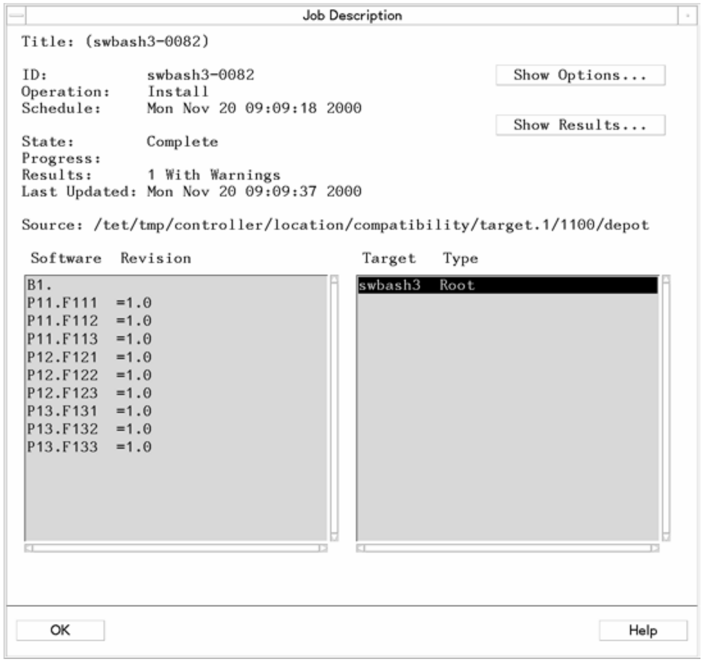
NOTE For performance reasons, a maximum of 250 targets are listed at once. If there are more then 250 targets for the job, **Next** and **Previous** buttons appear to let you view groups of 250 targets. Showing only warnings or errors can reduce the number of targets displayed.

Showing Job Descriptions

Selecting **Actions**→**Show Job Description...** opens the Job Description dialog, which contains all the information specified when the job was created, including:

- All job attributes (ID, type of operation, scheduling, current state, results, when last updated, source)
- Name and revision of the software
- Targets involved and target type:
  - — primary root
  - — alternate root
  - — depot

Figure 7-13 **Show Job Description**



```
Title: (swbash3-0082)

ID:           swbash3-0082
Operation:    Install
Schedule:     Mon Nov 20 09:09:18 2000

State:        Complete
Progress:
Results:      1 With Warnings
Last Updated: Mon Nov 20 09:09:37 2000

Source: /tet/tmp/controller/location/compatibility/target.1/1100/depot

 Software  Revision                      Target   Type
B1.                                     swbash3  Root
P11.F111  =1.0
P11.F112  =1.0
P11.F113  =1.0
P12.F121  =1.0
P12.F122  =1.0
P12.F123  =1.0
P13.F131  =1.0
P13.F132  =1.0
P13.F133  =1.0
```

- Selecting **Show Options...** displays the Job Options dialog. This lets you see the options used to create this job.
- Selecting **Show Results...** displays the Job Results dialog, which shows the latest status information on the job.
- Select **OK** in the Job Description dialog to return to the Job Browser. You can display the same information by double clicking on a job.

Showing Job Logs

Selecting **Actions→Show Job Log...** displays the Job Log dialog, which displays the controller (summary) log of a selected job. Buttons let you refresh or print the log file. Select **OK** to return to the Job Browser. (This menu item is greyed-out if the selected job is not active or completed.)

Copying Jobs

Copying a job consists of making the target and software selections available to a new session of swinstall, swcopy, or swremove. The new session is invoked automatically, using the same hosts, sources, software and target selections from the selected job. You can then re-use the same settings or make changes as needed.

This feature gives you the same advantages as using a session file in swinstall, swremove, or swcopy session. This can help you:

- Distribute the same software to a new set of targets
- Distribute new software to a previously defined set of targets
- Change a job from preview to full execution

To copy a job:

1. Select the desired job icon or listing in the job list.
2. Choose **Actions→Create Job from Selected Job**....

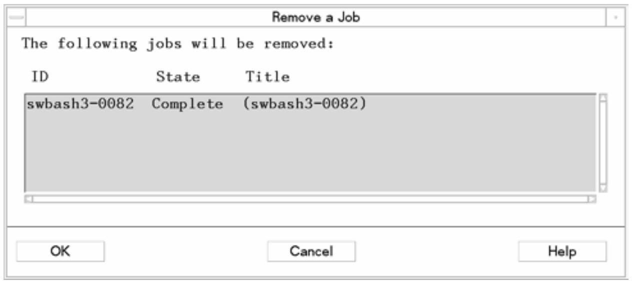
   The SD-UX program that matches the original job is automatically invoked. (For example, an installation job invokes swinstall.)

   - The Target Selection window displays the previously specified targets.
   - The Software Selection window displays the list of previously specified software.
3. Execute the program with the copied settings, or change the settings before execution.

Removing a Job

The **Actions**→ **Remove a Job...** menu choice lets you remove the currently selected jobs from the Jobs List. (To select more than one job, hold down the **CTRL** key while selecting jobs in the Job Browser window.) The Remove a Job dialog displays, listing information about the selected jobs.

Figure 7-14 **Remove a Job dialog**



```
The following jobs will be removed:

 ID            State     Title
swbash3-0082  Complete  (swbash3-0082)
```

- If you remove a scheduled job that has not yet run, the job is never run.
- You cannot remove a job that is in progress.

Monitoring Jobs from the Command Line

The swjob command lets you display and monitor jobs information using the command line. This command provides a quick, low-bandwidth alternative to the Job Browser when you want to check on specific jobs.

Syntax

```
swjob [XToolkit Options][-i][-R][-u][-v][-a attribute]
[-C session_file][-f software_file][-s source]
[-S session_file][-t target_file][-x option=value]
[-X option_file][software_selections][@ target_selections]
```

Options and Operands

XToolkit Options	X window options for use with `swjob -i`. See “XToolkit Options and Changing Display Fonts” on page 53.
`-i`	Starts the sd interactive job browser. See “Using the Job Browser” on page 217.
`-u`	Causes swjob to remove the specified jobs.
`-v`	List all available job attributes, one per line.
`-a` *`attribute`*	Display a specific attribute for the job. See “swjob Tasks and Examples” on page 233 for more information.
`-C` *`session_file`*	Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.
`-f` *`jobid_file`*	Read a list of job IDs from a separate file instead of (or in addition to) jobs you specify on the command line. (Use these files just like software files. See “Software Files” on page 58.)
`-S` *`session_file`*	Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-t` *`target_file`*
: Read a list of target selections from a separate file instead of (or in addition to) those you specify on the command line. See “Target Files” on page 59.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Using Command Options” on page 59.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Using Command Options” on page 59.

`jobid`
: One or more identification numbers for an SD-UX operation. You can read job ID numbers from the Job Browser when you set up or monitor your jobs interactively.

`target_selections`
: The target of the command. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swjob.

Table 7-3 **swjob Command Options and Default Values**

• admin_directory=/var/adm/sw • agent_timeout_minutes=10000 • log_msgid=0 • one_liner={jobid operation state progress results title} • rpc_binding_info= ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]	• loglevel=1 • match_target=false • rpc_timeout=5 • run_as_superuser=true • targets= • verbose=1

For More Information

See Appendix A, “Command Options,” on page 421 for more information about setting options and a complete listing and description of each option.

swjob Attributes

Each job has its own set of attributes. These attributes include job title, date of scheduled execution, and results. The `-a` option selects a specific attribute to display. You can specify multiple `-a` options to display multiple attributes.

Table 7-4 **Typical job attributes**

`jobid`	**The job identification number assigned by SD-UX**
`operation`	The type of operation (install, copy, remove, verify, etc.)
`state`	Current job status (completed, scheduled, or active)
`progress`	Number of targets completed
`results`	Completion status, indicating success, warnings, or errors
`title`	A title specified for the job by using the `job_title` option in swconfig, swcopy, swinstall, swremove, or swverify (see “Managing and Tuning Jobs with Command Options” on page 234)
`schedule`	Date at which the job was scheduled to run
`lastupdate`	Date at which information for this job was last updated

swjob Tasks and Examples

To simply list the jobs available, type:

```
swjob
```

To display attributes of jobs on the local system, type:

```
swjob -v
```

To display attributes of jobs on remote system `swbash3`, type:

```
swjob -v @ swbash3:/var/spool/sw
```

Using the `-a log` option lets you display log files for jobs. A job log file summarizes job details and target actions. For example, to display the depot log file for the job `swbash3-0008` on remote system `swbash3`:

```
swjob -a log swbash3-0008 @ swbash3:/var/spool/sw
```

NOTE You cannot specify any other `-a` options in the same command if you use `-a log`.

To remove job information for all previous jobs:

```
swjob -u \*
```

Managing and Tuning Jobs with Command Options

SD-UX command options let you manage and tune job behavior to best fit your environment, particularly when you run large numbers of jobs.

See “Using Command Options” on page 59 for additional information on setting command options.

Scheduling Jobs from the Command Line

The `-Q` *date* option lets you schedule jobs without starting the Job Browser. This option is available for swinstall, swcopy, swconfig, swremove, and swverify.

The format for *date* is:

```
MM/DD[/YYYY][,HH:MM][AM|PM]
```

For example, to install the C and Pascal products at 3 p.m. on June 23, 2001, at 10:14 a.m.:

```
swinstall -Q 06/23/2001,10:15AM -s sw_server cc pascal
```

Adding Job Titles

Purpose: to help you identify jobs.

When running large numbers of jobs, you may want to add more information to help you identify a specific job. You can do this from the Job Browser or by invoking swconfig, swcopy, swinstall, swremove, and swverify with the `job_title` command option. This lets you add an ASCII string that will be displayed along with the ID and other job attributes when you invoke swjob or the job browser.

For example, to install the C and Pascal products from depot `sw_server` to three remote hosts with a job title of `02-HLLs`:

```
swinstall -s sw_server -x job_title=02-HLLs cc pascal \
          @ hostA hostB hostC
```

Removing Job Information

Purpose: to increase performance and free up disk space when you run very large numbers of jobs.

SD-UX stores small amounts of information (such as job status or controller or agent logfiles) about each job. You can display this information from the Job Browser or with swjob. Keeping very large numbers of job information files may affect performance and reduce the usability of the Job Browser.

Running swconfig, swcopy, swinstall, swremove, or swverify with the `autoremove_job` command option set to true prevents SD-UX from storing the job information. (The trade-off is that you can not view the job information except by viewing log files.)

```
swinstall -s sw_server -x autoremove_job=true \
          MySoftware @ hostA hostB hostC
```

You can also use the -u option to remove all old job information:

```
swjob -u \*
```

You can also remove individual jobs from the Job Browser (see "Removing a Job" on page 229.)

8 Reliability and Performance

This chapter describes the interrelationship of Software Distributor reliability features and performance. Understanding how these features work together will help you improve the overall reliability of your software distribution system.

Table 8-1 Chapter Topics

Topics:
"Overview" on page 238
"Groups and Source Options" on page 240
"Large Numbers of Targets" on page 241
"Timeout Options" on page 242
"Retry RPC and Retry Interval" on page 243
"Retry Command" on page 244
"Database Checkpointing" on page 245
"Compression" on page 246
"Staging" on page 247
"Recovery (Install Only)" on page 250
"Installation With Separate Configuration" on page 252
"Multiple Versions" on page 253

Overview

SD-UX install and copy throughput are dependent on the following factors:

- Speed of the network
- Size (i.e., number of bytes) of the product being transferred
- Number of files being transferred
- Number of targets any one source is serving simultaneously

SD-UX provides many features that can be used together to increase the speed and success rates of distributed installations and copies. Many features are controlled by options set in the system defaults file, separate options file, command line, or GUI. This lets you set these features and options to best suit your own particular network and software needs.

These options and features can be categorized as follows:

- **Group and source options:** SD-UX eliminates the need to duplicate specification of commonly used groups of targets and software. Also, using the source option to specify a main depot reduces the number of dialog boxes.
- **Large numbers of targets:** Options that limit the number of simultaneous targets.
- **Timeout options:** Options that control how long the task attempts to retry low-level communications for file transfers before giving up.
- **Retry RPC and retry intervals:** Options that control the intervals between retries when the controller or targets attempt to re-establish lost connections.
- **Retry command:** Options that facilitate retrying operations that have failed. These can be used in conjunction with the checkpointing features or can start the task from the beginning.
- **Database checkpointing:** SD-UX commands perform automatic checkpointing at the fileset level and recording database transactions in the SD-UX depot catalog IPD. In addition, checkpointing at the file level is supported through attributes stored with the file.

- **Compression:** SD-UX supports compression functionality to reduce the amount of data being transferred.
- **Staging:** SD-UX supports staging software to intermediate depots either on the installation target or onto a source more accessible to the target, which has been preconfigured to use the alternate source.
- **Recovery:** SD-UX supports automatic procedures to recover from failed installations, leaving the system in the same state as previous. There are also manual means through use of multiple versions.
- **Multiple Versions:** SD-UX supports installing multiple revisions of the same software on a system at the same time, if the software supports it. Then, the old version can be unconfigured, and the new version configured as the active version. In case of any problems, the old version can be restored as the active version by unconfiguring the new version and reconfiguring the old.

Each of these topics is discussed in the following sections.

Groups and Source Options

Group and source options can help increase performance of some commands.

Target and software selections can be saved as group files and re-used. This reduces the need to re-specify commonly used selections, which reduces the time required to perform swinstall, swcopy, or swremove commands. See “Add/Save Software Group” on page 49, and “Software and Target Lists” on page 192 for more information.

Specifying source depots in advance can reduce the number of screens needed to run or schedule a swcopy or swinstall command. If you set the `source=` default option to your main source depot, SD-UX can immediately list the products available. This eliminates the Source Selection dialog, except when changing the default.

Large Numbers of Targets

The `max_targets` option applies to swinstall and swcopy operations. This option lets you manage hundreds of targets with a single job by limiting the number of simultaneous targets to a defined value. As each target completes the install or copy, another target is selected and started until all targets have been completed. This keeps the number of active operations at or below the user-defined limit.

The result of this option is that you can potentially manage hundreds of targets reliably in a single task. This can also create a significant reduction in the time it takes to schedule a task.

Your server and network performance will determine the optimal setting for this option. The default setting is 25.

Timeout Options

Timeout options control how long a task continues to retry low-level communications for file transfers before giving up.

One control is the amount of time each single RPC call waits before giving up. This timeout is set by the `rpc_timeout` option. Legal values are 0 through 9. The default value is 5, which corresponds to about 30 seconds for the UDP protocol. Each value doubles the time of the preceding value (i.e., a value of 4 is about 15 seconds).

Another control is the number of times a target agent attempts to reconnect to the source agent after an `rpc_timeout` has detected a lost connection source during an installation of a fileset. The number of retry attempts is controlled by the `retry_rpc` option. The range of values is 0 through 9. A 0 value means no retry is attempted; if the connection is lost, the command will fail. The default value is 1. For less stable networks, a value of 5 is recommended.

NOTE When setting `retry_rpc` to a value greater than 0, the `reinstall_files` option should also be set to `false` (the default value), so the same files are not recopied when a fileset is retried.

The maximum possible amount of time spent waiting for the timeout is affected by a combination of `rpc_timeout`, `retry_rpc`, and `retry_rpc_interval`.

For troubleshooting information on Timeout options, refer to “Connection Timeouts and Other WAN Problems” on page 472.

Retry RPC and Retry Interval

During a swinstall or swcopy operation, `retry_rpc_interval` controls the interval schedule for repeated attempts to make a connection to a target or the source agent after an initial failure. This option works in conjunction with `retry_rpc`, which controls the number of times the target or source is re-contacted.

The default value for `retry_rpc` is 1 and `retry_rpc_interval` is {0}.

The recommended values for the `retry_rpc_interval` algorithm are {1 2 4 8 15} with `retry_rpc` set at 5. If the agent session fails to start for any reason, the controller and/or target will attempt to re-connect in the following way (i.e., both options set as above):

- after 1 minute for the first try
- then 2 minutes for the second try
- then 4 minutes
- then 8 minutes
- finally 15 minutes for the last try

If the number of values for `retry_rpc_interval` is less than `retry_rpc`, the last value given is repeated until the number of actual retries is equal to `retry_rpc`.

NOTE If `retry_rpc` has a value of zero, no retry is attempted. An initial value of 5 is recommended for WAN environments.

Retry Command

SD-UX supports options that facilitate retrying operations that have failed. These can be used in conjunction with the checkpointing features or can start the task from the beginning.

Each execution of a command records all of the target, software, and option selections automatically in a session file, *command*.last. You can also save the session information to a different file using the GUI. The session files are stored in the directory `$HOME/.sw/sessions`. The entire command can be retried by recalling the session in the GUI or by re-executing the task:

```
command -S command.last
```

When a task is retried, any fileset that is up to date (has the same product and filesets revisions as available or installed and is not in the *transient* or *corrupt* state) will not attempt to be reinstalled (the default behavior), and all other filesets will be retried. This behavior can be overridden, forcing the retry to start at the beginning of all files, by setting `reinstall=true` from either your option setting or the CLI.

If the `reinstall_files` option is set to true, all files in that fileset are retransferred. However, although preinstall and postinstall scripts for filesets that are being installed are executed normally, the file transfer for up to date files can be avoided by leaving the `reinstall_files` option equal to false (the default).

When the `reinstall_files` option is false, the user can also control which attributes are checked in order to determine if the fileset is already installed or available. If the `reinstall_files_use_cksum` option is set to true, the size, mtime, and cksum attributes are checked.

If the `reinstall_files_use_cksum` option is false, then only the size and mtime are checked. Checking the cksum attribute is more time consuming but more reliable. The size and mtime checks are very fast.

The user can see which files were actually installed or copied and which were skipped due to being already up to date by setting the `loglevel` option to 2.

Database Checkpointing

The tools perform automatic checkpointing, recording transactions in the SD-UX depot catalog, or Installed Products Database (IPD) at the fileset level. Additionally, checkpointing at the file level is supported through attributes stored with the file.

During a swinstall or swcopy operation, all filesets in the current product being loaded are recorded in the depot catalog or IPD as having a state of transient. After all filesets in a product complete the copy or install, the state is changed to available or installed, and the next product is started. At this point, retrying an operation will not attempt to recopy or reinstall the filesets that are already installed (see “Retry Command” on page 244).

NOTE This behavior requires that either the product or fileset have a revision defined.

The current state and revisions of filesets can be displayed with the command:

```
swlist [-d] -l fileset -a revision -a state
```

If there is an error installing a fileset in the product that causes the install to fail (e.g., lost connection to the source), all filesets in the product are changed from transient to corrupt. (All filesets are assumed corrupt since the product level postinstall script has not been run yet. In actuality, the filesets may be properly installed.)

Independent of a fileset being installed (either properly or in a *corrupt* state) you can determine whether any particular file is installed properly with a high degree of certainty through the file’s size, mtime, and cksum attributes. Through these file attributes, checkpointing at the file level is approximated (this is described in the previous section).

Compression

The swinstall and swcopy commands can transfer large amounts of data over the network from depots to targets. The SD-UX `compress_files` option can improve performance by first compressing files that are to be transferred. This can reduce network usage by approximately 50%; the exact amount of compression depends on the type of files. Binary files compress less than 50%; text files generally compress more.

Set this option to true only when network bandwidth is clearly restricting total throughput. If it is not clear that this option will help, compare the throughput of a few swinstall or swcopy tasks (i.e., with and without compression) before changing this option value.

You can use swcopy to compress files and leave them compressed in a target depot or compress before network transfer and uncompress afterward.

Precompressing a depot is advantageous when installing or copying to multiple targets. If the source depot is not already compressed, then each file is recompressed for each target.

You can set `uncompress_files` to true to leave a depot uncompressed after copying with swcopy. For swinstall, the `compress_files` option will compress all uncompressed files before network transfer. Files are always uncompressed before installing them to the target file system.

INDEX and INFO Compression

Another way to reduce your network traffic is by compressing INDEX and INFO files from the source depot to the target. You can turn on INDEX or INFO compression by setting the `compress_index` option to *true* in the defaults file (`/var/adm/sw/defaults`).

The SD-UX controller and target agents will request compressed INDEX files from the source agent. If the source agent is read only or an older version of SD-UX, the agent cannot comply; consequently, the client will request a normal INDEX. Otherwise, the source agent will send a precompressed INDEX and INFO or compress it on the fly.

The target agent will then create a permanent compressed INDEX in the target, depot, or root. This saves the next request for a compressed INDEX or INFO from having to compress on the fly.

Staging

The standard way to install software onto multiple targets is to specify a single source depot and each target that is to receive the software. However, some software distribution environments require that you manage software on large numbers of geographically dispersed target systems. This may require the use of one or more intermediate source depots or staging areas. This variant on the standard model is referred to as a **staged installation**.

There are two reasons for using a staged installation:

1. Minimize the amount of data transferred across a slow and expensive segment of your network.
2. More easily ensure a successful installation on all targets by reducing the risk of an unreliable segment in your network.

If your environment has targets organized in separate, local area networks (LAN) and connected via a low-throughput, less-reliable wide area network (WAN), staging software to intermediate depots that are local to each grouping of targets and then doing the installation using these intermediate depots reduces the amount of data that travels over the WAN segment.

By doing so, you also decrease the likelihood that a problem with the WAN will interrupt the installation step.

Before you do a staged installation, you must first decide where the intermediate depots should reside. Here are two possible approaches:

1. If the targets are grouped, you can put an intermediate depot on one system in each group and configure the other targets to use it as their alternate source. This approach requires that each target in the group be configured to use the designated intermediate depot.
2. If making sure that installations succeed is of highest importance, you can locate the intermediate depots on the targets themselves, one-per-target. An advantage to this approach is that it doesn't necessarily require that you configure an alternate source on each target. However, this approach requires that each target system have enough disk space to accommodate the intermediate depot.

To do a staged installation:

1. First, decide on the location of the intermediate depots and use the swcopy command to copy the software from your master depot to them. This step is no different from a normal multi-target copy operation.

```
swcopy -s master -t depot_list NewApp
```

   In this example, the master source depot containing the product `NewApp` is in the default `/var/spool/sw` depot location and a file named `depot_list` contains the list of intermediate depots.

   The `depot_list` could identify the designated intermediate depots that have been configured for each group of targets, or it could identify an intermediate depot located on each target.

2. Next, use the swinstall command combined with the option `use_alternate_source=true` to do the actual installation. The `use_alternate_source` option is specified from either the CLI (i.e., `-x use_alternate_source=true`) or via the Options Editor window in the GUI. The default value is false.

```
swinstall -s master -x use_alternate_source=true \
     -t targ_list NewApp
```

   The `use_alternate_source=true` option instructs each target to use its own configured source for the installation. The source that is specified on the swinstall CLI is used only by the controller for the validation of your software selections. The file `targ_list` contains the list of targets.

   When `use_alternate_source` is true, each target agent looks for the corresponding `swagent.alternate_source` option in its own defaults file. The protocol sequence and endpoint given by the option, `swagent.rpc_binding_info`, are used when the agent attempts to contact the depot specified by `swagent.alternate_source`. An alternate source is specified using the *host:/path*, */path*, or *host* syntax.

   - If there is a *`host:/depot_path`* specified in the target's `swagent.alternate_source` option, the agent gets the software from this source. If only a *`host`* is specified, the target agent uses the same depot path used by the controller.

- If the target doesn't have an alternate source, the agent uses the same depot path used by the controller, but it will apply this path to its own file system. This lets you do staged installations without any target configuration at all, by locating the intermediate depot on each target system at the same file system location as the master depot (approach 2 above).

Because the swcopy and swinstall steps in a staged installation are separate, SD-UX cannot enforce consistency between master and intermediate depots. You must ensure that the software available from the intermediate depots is consistent with that on the master depot.

If master and intermediate depots are out-of-synch when you perform the swinstall step, you may encounter errors if software that is on the master depot is not available from one or more intermediate depots.

Recovery (Install Only)

NOTE This section applies only to customer-created software with unpreinstall and unpostinall scripts. HP-supplied software does not include these scripts.

SD-UX supports automatic procedures to recover from failed installation if the `autorecover_product` option is set to true, attempting to leave the system in the same state as it was previously. Also, manual means are available (refer to “Multiple Versions” on page 253).

Rollback is limited to the system where the installation of the product failed, not all target systems specified in an installation job.

Because autorecovery removes any files that were installed up to this point, it is antithetic with the checkpointing and retry features previously described. Recovery saves copies of each file that it is replacing, then removes those files at the successful completion of the product installation. If the install fails, then the saved files are restored.

Once a product is successfully updated, it cannot be restored except by reinstalling it. Additionally, if a later product fails, the earlier product cannot be recovered. In order to meet the requirement that multiple products be recoverable, multiple versions must be installed.

The unpreinstall and unpostinstall scripts are needed to undo the steps that the preinstall and postinstall scripts executed. The normal sequence of operations for each product is:

1. Execute the product preinstall script
2. For each fileset
   a. execute the preinstall script
   b. install the files
   c. execute the postinstall script
3. Execute the product postinstall script

If any of these steps fails (e.g., a lost source or a script error) then the undo scripts are run, and the files restored from the point of failure in reverse order.

NOTE Patches created using the features capabilities described may maintain saved files. In this case, patches can be removed (rolled back) or committed (by removing saved files). See Chapter 5, “Managing Patches,” on page 163 for more information on patches.

NOTE The use of `autorecover_product=true` during an update of the HP-UX OS is not supported.

Installation With Separate Configuration

NOTE Because deferring configuration of OS software and patches can leave the system in an unusable state, do not use this technique with HP-supplied software.

If you create your own software that includes configuration scripts to be performed automatically after installation, performing the configuration separately can increase the reliability of the overall installation process.

To install without configuring, set the `defer_configure` option to true for swinstall. Then, after all the installs have completed successfully, you can run the configure scripts for all targets at once by using the swconfig command.

Multiple Versions

SD-UX supports installing multiple revisions of the same software on a system at the same time, if the software supports it. By using multiple installed versions, recovery can be supported at the system or task (all systems) level.

Installing a second version requires some careful planning, as well as understanding how to identify multiple versions on the system.

Each product has a product directory attribute. The installed location on the target is by default the same as the product directory. For example, a product `Foo` might have a product directory of `/opt/foo`. You can list the locations of installed software with:

```
swlist -l product -a location
```

or

```
swlist -l product -a software_spec
```

A common practice is to install a second version of the product. When installing this software, a new location must be selected. In the case of the product `Foo`, the new location might be `/opt/foo.v2`. After specifying the new location (i.e., by adding `l=/opt/foo.v2` after the product tag in the GUI or CLI), swinstall will replace the product directory portion of all files with the new product location.

NOTE The `allow_multiple_versions` option must be set to true for swinstall to install multiple versions of the software. The new version will not be configured by swinstall if there is another version configured.

After a second version has been installed, each version can be identified either by the location (`Foo,l=/opt/foo` and `Foo,l=/opt/foo.v2`), by the revision (`Foo,r=1.0` and `Foo,r=2.0`), or both. You can list the locations and revisions of all versions with:

```
swlist -l fileset -a location -a revision
```

Additionally, you can list a *fully qualified software spec* containing both the location and revision as well as the other *version distinguishing* attributes (vendor and architecture) with:

```
swlist -l fileset -a software_spec
```

After the new version is installed successfully for all products or on all hosts, the old version can be unconfigured, and the new version configured as the active version by using **`swconfig -u`** with the old version and swconfig with the new version.

NOTE By default, only one version of the software is allowed to be configured at a time.

The second version can not be configured until the first one is unconfigured. As was discussed in “Installation With Separate Configuration” on page 252, installing a second version automatically excludes configuring the new version.

You must manually unconfigure the old version, then configure the new version. If the software supports multiple configured versions (in addition to multiple installed versions) the `swconfig.allow_multiple_versions` option can be set to `true`.

In case of any problems, the old version can be restored as the active version by unconfiguring the new version and reconfiguring the old (i.e., by **`swconfig -u`** with the new version and swconfig with the old version).

In order to support multiple versions, the software must be structured so that all files are below the product directory, and the configure scripts need to be written with multiple version support in mind. In a simple example, the configure script could add a symbolic link from `/usr/bin/foo` to `$SW_LOCATION/bin/foo`, and the unconfigure script could remove that link. In this example, configuring and unconfiguring each version of this software is easily done.

NOTE The use of `allow_multiple_versions=true` command option and the `l=<alternate location>` software specification is not supported when updating HP-UX to a new version.

9 SD-UX Security

During the SD-UX installation, a default security setup is created. This chapter explains basic SD-UX security, introduces the swacl command, presents examples of common tasks, and provides in-depth discussion of how SD-UX manages security.

Table 9-1 **Chapter Topics**

Topic and Page
"Overview" on page 256
"The swacl Command" on page 258
"Basic Security Tasks" on page 261
"How ACLs are Matched to the User" on page 272
"ACL Entries" on page 273
"Security on SD-UX Systems" on page 285
"SD-UX Internal Authentication" on page 287
"RPC Authorization" on page 291
"Security Use Models" on page 295
"Permission Requirements, by Command" on page 298

Overview

Along with the traditional HP-UX file access protection, SD-UX uses Access Control Lists (ACLs) to protect the primary objects on which it manages software:

- Hosts
- Roots (software installed on a host)
- Depots
- Products within depots

An ACL consists of a set of entries associated with an object when it is created.

Default Security

The following security scheme exists by default:

- The local superuser always has access to all local objects.
- Read access is provided to all users on the network who use the same SD-UX shared secret via the `any_other` ACL.
- Whoever creates a root, depot, or product object has full access to it as the `object_owner`.
- If you set up systems for remote operations (using the procedure discussed in “Setting Up Remote Operations” on page 199), `root@central_controller` has full access to all target objects via the `user:root@central_controller` ACL.

If you are running as `root@central_controller`, the suggested security setup should be adequate to perform all tasks.

Two templates are used to create default ACLs:

- `global_soc_template` (applies to all new depots and roots added to the host)
- `global_product_template` (applies for new products in depots)

Depots and Depot Registration

Software Distributor typically uses central depots to distribute software. You can control access to these depots by users who will install software.

An important security consideration is that depots *must* be registered for nonlocal users to have access. Only a local superuser or a user with insert permission on the host can install from unregistered depots.

For more information, see “Registering and Unregistering Depots (swreg)” on page 151 and “Depot Management Commands and Concepts” on page 134.

Modifying Target Systems

You may want to set up each system to grant administrative access to the SD-UX controller while restricting access to other systems and users.

You will need to modify ACLs on your target systems in the following cases:

- To change the login name of the SD-UX administrator (the default is root).
- To modify permissions for the SD-UX administrator or group of administrators.

The swacl Command

The swacl command lets you view or change ACL entries and permissions.

swacl Syntax

```
swacl ] -l level [-D acl_entry| -F acl_file| -M acl_entry]
    [-f software_ file] [-t target_ file]
    [-x option=value]  [-X option_file]
     [software_selections] [@ target_selection]
```

Options and Operands

`-l` *`level`* — Level to edit. Level designations are the literals: `host, depot, root, product, product_template, global_soc_template` or `global_product_template`. (See "ACL Templates" on page 282 for a complete discussion.)

NOTE You can change an ACL with `-D`, `-F`, or `-M` command options. You can only specify one of these options per command because they are mutually exclusive. If you don't specify a `-D`, `-F`, or `-M` option, swacl prints the specified ACLs.

`-D` *`acl_entry`* — Deletes an existing entry from the ACL associated with the specified object. You can enter multiple `-D` options.

`-F` *`acl_file`* — Assigns the ACL information contained in *`acl_file`* to the object. All existing entries are removed and replaced by the entries in the file. You can enter only one `-F` option.

`-M` *`acl_entry`* — Adds a new ACL entry or changes the permissions of an existing entry. You can enter multiple `-M` options.

`-f` *`software file`*

Reads a list of software selections from a separate file instead of from the CLI. (See "Software Files" on page 58.)

`-t` *`target file`*
: Reads a list of target host selections from a separate file instead of from the CLI. (See "Target Files" on page 59.)

`-x` *`option=value`*
: Lets you change an option on the command line interface (CLI) that overrides the default value or a value in an alternate options file (`-X` *`option file`*). See "Changing Command Options" on page 259.

`-X` *`option file`*
: Uses the option values in a specified *`option file`*. See "Using Command Options" on page 59.

`software_selections`
: The software objects for the swacl operation. See "Software Selections" on page 56.

`target_selections`
: The target of the command. See "Target Selections" on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the defaults and options that apply to swacl.

Table 9-2 swacl Command Options and Default Values

- admin_directory=/var/adm/sw
- distribution_target_directory= /var/spool/sw
- installed_software_catalog=products
- level=
- log_msgid=0
- rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]
- rpc_timeout=5
- run_as_superuser=true
- select_local=true
- targets=
- verbose=1

For More Information

See Appendix A, "Command Options," on page 421 for complete descriptions of each default.

swacl Output

A typical list output from the `swacl` command looks like the following:

```
# swacl    Installed Software Access Control List
#
# For host:  prewd:/
#
# Date:  Mon Nov 06 16:39:58 2001
#
# Object Ownership: User=root
#                   Group=sys
#                   Realm=prewd.fc.hp.com
# default_realm=prewd.fc.hp.com
object_owner:crwit
user:rml:crwit
user:root@newdist.fc.hp.com:crwit
group:swadm:crwit
any_other:-r--t
```

The header information (lines marked with #) gives the object's name and owner and the name of the user's **realm** or host name of the user's system. In this case:

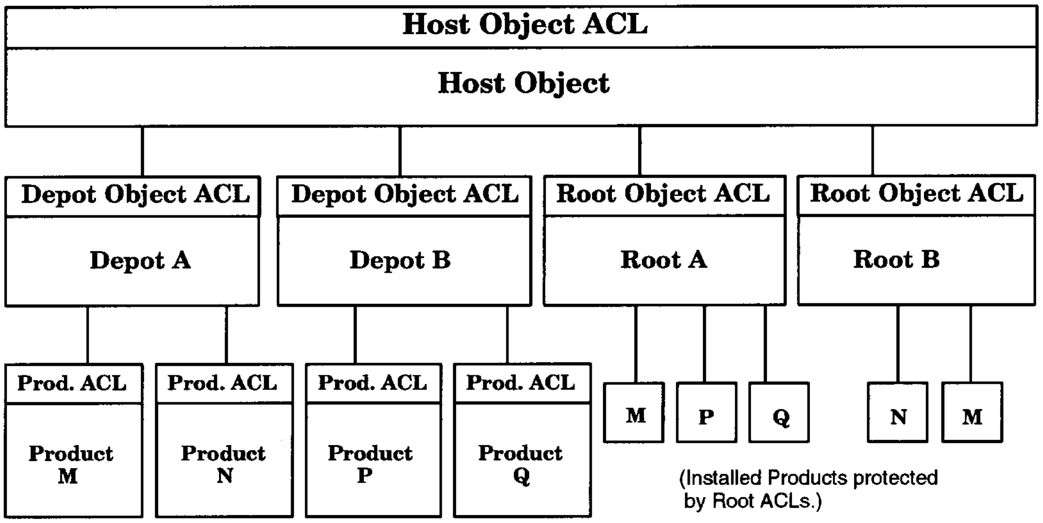
- The object is a root (installed software).
- The creator of the object (`object_owner`) is `root@prewd.`
- The `object_owner` has all permissions.
- Local user `rml` and remote user `root@newdist.fc.hp.com` have all permissions.
- Members of local group `swadm` have all permissions.
- Anyone not matching one of the previous entries (`any_other`) has read and test permissions.

For more information on the format of the ACL file, see "ACL Entries" on page 273.

Basic Security Tasks

Along with the traditional HP-UX file access protection, all SD-UX objects (hosts, depots, roots and products) are also protected by ACLs.

Figure 9-1 Access Control Lists



ACLs offer a greater degree of selectivity than do permission bits. An ACL extends the concept of the HP-UX file system's permission bits by letting you specify different access rights to several individuals and groups instead of just one of each.

For example, if you set up remote operations, you must install an HP ServiceControl Manager fileset that makes some elementary changes to the security ACLs on the remote systems. One of these changes is to copy three ACLs from the source system to the destination systems. (See "Setting Up Remote Operations" on page 199.)

The ACLs copied are those protecting the source host (the host ACL), the host's template ACLs used in subsequent operations to produce ACLs for products (the `global_product_template`), and depot/root containers (the `global_soc_template`). When copied, these ACLs grant users on the source host the same permissions on the destination host as they have locally on the source host. In addition, an entry for the superuser at the source host was added. This lets the controller system's superuser perform software distribution tasks on the remote system without having to reconfigure ACLs.

If you need to change security, the following tasks can be performed (i.e., to understand and modify the default setup):

- Listing user access
- Allow user to manage products in a depot
- Allowing users to manage roots
- Restricting read access to a depot
- Adding target hosts
- Temporarily restricting access to a depot
- Closing the SD-UX network
- Editing an ACL

Listing User Access

The following examples show how to list users with access to depots, targets host, target root, and all products.

- Display the default root ACLs on a newly installed HP-UX 11i system:

  `swacl -l root`

  ```
  #
  # swacl  Installed Software Access Control List
  #
  # For host:  swelter:/
  #
  # Date:  Wed Feb 28 14:58:02 2001
  #
  # Object Ownership:  User= root
  #                    Group=sys
  #                    Realm=swelter.fc.hp.com
  #
  # default_realm=swelter.fc.hp.com
  object_owner:crwit
  any_other:-r---
  ```

This ACL indicates that the file system is owned by the root user, and that as such, the owner has full ACL permissions (`crwit`). Additionally, all other users may read SD information about this root file system using the swlist command.

- To list the users with access to the default depot (optionally on a remote host):

  `swacl -l depot @ newdist`

```
#
# swacl    Depot Access Control List
#
# For depot:  newdist:/var/spool/sw
#
# Date:  Fri Nov 03 11:23:51 2001
#

# Object Ownership:  User= root
#                    Group=other
#                    Realm=newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
user:rmr:crwit
user:root:crwit
user:fred@hpfred.fc.hp.com:crwit
user:root@hpfcpsm.fc.hp.com:crwit
user:root@wookie.fc.hp.com:crwit
any_other:-r---
```

- To show access to installed software:

  `swacl -l root @ newdist`

```
# swacl    Installed Software Access Control List
#
# For host:  newdist:
#
# Date:  Fri Nov 03 10:33:04 2001
#
# Object Ownership:  User= root
#                    Group=other
#                    Realm=newdist.fc.hp.com
```

```
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
user:root:crwit
user:root:crwit
any_other:-r---
```

- To show permission to create depots and roots on the target host:

  `swacl -l host @ newdist`

  ```
  #
  # swacl    Host Access Control List
  #
  # For host:  newdist
  #
  # Date:  Fri Nov 03 10:34:06 2001
  #

  # Object Ownership:  User= root
  #                    Group=sys
  #                    Realm=newdist.fc.hp.com
  #
  # default_realm=newdist.fc.hp.com
  user:fred:crwit
  user:root:crwit
  user:smp:crwit
  user:root@udltools.fc.hp.com:crwit
  user:fred@hpfred.fc.hp.com:crwit
  user:chrisr@prewd.fc.hp.com:crwit
  any_other:-r---
  ```

- To list the users with access to all products (“*”) in a depot:

  `swacl -l product \* @ newdist:/var/spool/sw`

  ```
  #
  # swacl    Product Access Control Lists
  #
  # For depot:  newdist:/var/spool/sw
  #
  # Date:  Fri Nov 03 10:34:06 2001
  #

  # For product:  product1,r=1.0
  ```

```
#
# Object Ownership:  User= root
#                    Group=other
#                    Realm=newdist.fc.hp.com
#
# default_realm=newdist.fc.hp.com
object_owner:crwit
user:root:crwit
user:root@prewd.fc.hp.com:crwit
any_other:-r---
```

Allowing Users to Manage Products in a Depot

Users that are packaging products may need access to the SD-UX depots to store their products.

In ACLs, **a** is a shorthand notation for all permissions (`crwit`).

To allow user `mary` to add new products to the depot:

`swacl -l depot -M user:mary:a [@` ***`host:depot`*** **`]`**

To allow access for user *`mary`* to modify all existing products in a depot:

`swacl -l product -M user:mary:a \* [@` ***`host`*** **`]`**

To modify the template so that user `mary` can modify new products created by others in the depot:

`swacl -l global_product_template -M user:mary:a [@` ***`host`*** **`]`**

(In the above examples, change `user` to `group` and use a group name to add group access to the depot structures.)

Allowing Users to Manage Roots (Install/Remove)

To give a user (`mary`) the necessary permissions to be able to install or remove software on host `mysys`:

`swacl -l root -M user:mary:a @ mysys`

To allow user mary to install software into the default root:

`swacl -l root -M user:mary:ri`

To give user mary the permission to open the root for reading:

`swacl -l root -M user:mary:r`

To give user mary the permission to install new software into the root object:

```
swacl -l root -M user:mary:i
```

To let remote user `allen@swelter` fully manage the root file system on `swcrunch`:

```
swacl -l root -M user:allen@swelter:a
```

(In the above examples, change `user` to `group` and use a group name to add group access to the depot structures.)

NOTE Because software installation usually involves modification of system files during configurations, software install and configure scripts are run as the superuser. Therefore, granting a user write permission on a root is essentially giving them superuser access for managing software.

Restricting Access to Depots

To restrict read access to a depot you must first remove `any_other` access from the depot *and* from the products contained in the depot *and* the template controlling the products in the depot.

You can restrict access to depot `alpine` on host `drgw`:

```
swacl -l depot -D any_other @ drgw:/alpine
swacl -l product -D any_other \* @ drgw:/alpine
swacl -l global_product_template -D any_other \* \
      @ drgw:/alpine
```

You will then need to add specific users (and then hosts) with read access after removing `any_other` from the depot security. The following commands add `read` access for any user on `hostA` to the depot, the products contained in the depot, and future products, respectively.

```
swacl -l depot -M other:@hostA:r @ drgw:/alpine
swacl -l product -M other:@hostA:r \* @ drgw:/alpine
swacl -l global_product_template -M other:@hostA:r \
      @ drgw:/alpine
```

In the following example, the local superuser disallows all remote users from accessing `/simple_1.depot` on `swelter`, but allow local users to access the depot:

```
swacl -l depot -D any_other @ /simple_1.depot
swacl -l depot -M other:r @ /simple_1.depot
swacl -l depot @ /simple_1.depot
```

```
#
# swacl    Depot Access Control List
#
# For depot:  swelter:/simple_1.depot
#
# Date:  Thu Mar  1 16:19:57 2001
#
# Object Ownership:  User= allen
#                    Group=users
#                    Realm=swelter.fc.hp.com
#
# default_realm=swelter.fc.hp.com
object_owner:crwit
other:-r---
```

Local users can now access this depot as a result of the `other` ACL, but remote users are refused.

To allow only user `shelly` on host `swcrunch` to access software in a depot located on `swelter`, it may appear that adding a user ACL for shelly would be sufficient:

```
swacl -l depot -M user:shelly@swcrunch:r @ /simple_1.depot
```

However, this is not enough. An attempt by `shelly` to access this depot would fail with a security violation. This is because SD-UX also requires that SD agents (the swagent process) that contacts the depot server to be authorized via a `host` ACL `entry_type`:

```
swacl -l depot -M host:swcrunch:r @ /simple_1.depot
```

(Note that user `shelly` also requires appropriate ACL permission to install software on `swcrunch`.)

NOTE The **r** (read) permission allows the user to access the depot and products, and the **t** (test) permission allows the user to list the ACLs.

Adding Target Hosts

For swinstall and swcopy, both the user and target host are validated (i.e., to protect from unauthorized users at remote hosts switching to an authorized user). The following adds read permission for the host named `target` to the default depot on the local host, the products currently in the depot, and any future products added to the depot (using `global_product_template`).

```
swacl -l depot -M host:target:r
swacl -l product -M host:target:r \*
swacl -l global_product_template -M host:target:r
```

Since the user is always validated, another alternative that makes it easier to manage large numbers of hosts is to allow all hosts read permission:

```
swacl -l depot -M host:*:r
swacl -l product -M host:*:r \*
swacl -l global_product_template -M host:*:r
```

NOTE "*" is only a supported value for the *host* ACL type.

Temporarily Restricting Access

A simple method of restricting access to anyone other than the local superuser without modifying ACLs is to unregister the depot.

```
swreg -u -l depot [@ depot]
```

It can then be reregistered later:

```
swreg -l depot [@ depot]
```

Closing the SD-UX Network

The SD-UX **secret** is used a a proof of trustworthiness for the caller's credentials. It is a password that SD-UX uses to verify the authenticity of the caller's host. The default secret field is set by manufacturing to match the default setting on the HP-UX controller. All secrets (i.e., controller, targets, and depots) *must* be identical.

NOTE Do not change the default secret field unless you have also changed the default secret on the HP-UX SD-UX controller. These two secrets must match.

The set of hosts that can be managed by SD-UX can be restricted by changing the default secret on all SD-UX controller and target hosts in the network. The default secret is found in `/var/adm/sw/security/secrets`.

You may change the default secret found in this file:

default *`new secret`*

For additional information, see “Security Between Hosts: The Shared Secrets File” on page 289.

Editing an ACL

The swacl command, when invoked without the `-M`, `-D,` or `-F` options, reads the specified ACL, converts it into plain text and prints it to `stdout`. The output of the command can also be redirected to a file, which can then be printed or edited. After editing, you can use the `-F` *`file`* option described above to replace the entire old ACL. This procedure gives you full ACL editing capabilities.

You must have test permission within the ACL to produce the edit file (list the ACL) and control permission to modify it with `-F`, `-D`, or `-M` options. All ACL entries must contain test permission.

If the replacement ACL contains no detectable errors and you have the proper permission on the ACL, the replacement will succeed. If the replacement fails because you lack permission to make the change, an error is generated, and the object is skipped.

You may change or delete existing entries, or you may add additional entries to the ACL.

NOTE It is possible to edit an ACL so that you cannot access it! Caution should be used to avoid accidentally removing your own control (c) permissions on an ACL. As a safeguard, the local superuser may always use swacl to edit SD-UX ACLs.

Here are some examples based on the following ACL that is protecting a product (`FORTRAN`) created by user `rob` whose local host is `lehi.fc.hp.com`:

```
# swacl    Product Access Control Lists
#
# For host:  lehi:/
#
# Date:  Mon Nov 06 16:39:58 2001
#
# For product: FORTRAN,r=9.0,v=HP
# Object Ownership:   User=root
#                     Group=sys
#                     Realm=lehi.fc.hp.com
# default_realm=lehi.fc.hp.com
object_owner:crwit
user:barb:-rt
user:ramon:-rt
group:swadm:crwit
host:alma.fc.hp.com:-rt
any_other:-rt
```

You can list the ACLs for the product is `FORTRAN` in depot `/var/spool/sw` (the default depot) and prepare it for editing:

```
swacl -l product FORTRAN >acl_tmp
```

This will bring the above ACL into the file `acl_tmp`, and it is ready for editing. Edit the `acl_tmp` file with any suitable text editor.

To replace all entries in the ACL for `FORTRAN`, type:

```
swacl -l product -F acl_tmp FORTRAN
```

To edit the default product template on a depot `/var/spool/sw_dev`, use:

```
swacl -l product_template @ /var/spool/sw_dev >tmp_file
```

Then edit the `tmp_file` and replace the ACL:

```
swacl -l product_template -F tmp_file \
      @ /var/spool/sw_dev
```

To delete entries for user `barb` and group `swadm`, use:

```
swacl -D user:barb  -D group:swadm -l product FORTRAN
```

To give user `ramon` permission to modify the product `FORTRAN`, type:

```
swacl -M user:ramon:trw -l product FORTRAN
```

To add an entry for user `pam` with complete management permission (“a” is shorthand for `crwit`), use:

```
swacl -M user:pam:a
```

To add an entry to grant every user in group `swadm` at remote hosts `dewd` and `stewd` full management control of the product `FORTRAN` on the default local depot, use the following:

```
swacl -M group:swadm@dewd:a -M group:swadm@stewd:a \
      -l product FORTRAN
```

To list the ACL protecting the default depot at host `dewd`, type:

```
swacl -l depot @ dewd
```

How ACLs are Matched to the User

ACL permissions are determined by a match to a *single* ACL entry, not to an accumulation of matching entries. Checking is done from the most restrictive entry types to the broadest.

If a match is found in a user entry type, no further checking is done, and the permissions for that user are fully defined by the permissions field of the matched entry. A matched user may be a member of a group with broader permissions; this has no consequence.

NOTE The local superuser has access to all local SD-UX objects irrespective of ACLs.

The ACL matching algorithm is:

1. If user is local superuser, then grant all permissions.
2. If user is owner of the object, then grant `object_owner` permissions.
3. If user matches a `user` entry, then grant `user` permissions.
4. If any `group` entries match, then accumulate the permissions granted by all group entries that match the user's primary and supplementary groups.
5. If an appropriate other entry matches, then grant other permissions.
6. If an `any_other` entry, then grant `any_other` permissions.
7. Grant no permissions.

ACL Entries

An ACL consists of a set of **entries** attached to an object when it is created. These entries define which users, groups, and/or hosts have permission to access the objects. ACL entries include the concept of a **principal**, which is the user, group or host system (for agents making RPCs) that originates a call to another system.

An ACL entry consists of three fields:

entry_type[:*key*]:*permissions*

For example, an ACL entry for an SD-UX object might be:

`user:fred:r-ctw`

This means that a user named `fred` can `control (c)`, read (`r`), write (`w`), and test (`t`) the object, but the dash signifies that he cannot `i` (insert/create) new objects.

NOTE You can specify `crwit` permissions in any order.

The ACL `entry_type` must be one of these values:

Table 9-3 **SD-UX ACL Entry Types**

Type	Permissions Apply To
`user`	User principal, whose name is to be specified in the `key` field
`group`	Group principal, whose name is to be specified in the `key` field
`host`	Host systems (target agents acting on behalf of users for install or copy)
`other`	Principals with no matching user and group entries

Table 9-3 **SD-UX ACL Entry Types (Continued)**

Type	Permissions Apply To
`any_other`	Principals not matching any other entry
`object_owner`	Owner of the object
`object_group`	Members of the group to which an object belongs

Do not confuse the host object (which is a computer system that contains depots, roots, and software) with the `host` *entry type (which defines permissions for access to target systems).*

The user and group of the object's owner are determined and automatically recorded at the time the object is created (based on the identity of the person who creates it). This information is recorded as `user`, `group`, and `realm`. An `object_owner` or `object_group` entry type in an ACL causes the SD-UX ACL manager to look up the owner and group information on the object; and if a match to the requester is found, grant permissions as specified.

There may be many `user`, `group`, and `host` type entries per ACL, while there may be only one of each of `object_owner`, `object_group` and `any_other`. There may be at most one local (i.e., no key) other entry and an unlimited number of remote (i.e., keyed) other entries.

ACL Keys

The second part of the ACL entry is the **key**. The table below lists the possible key values for specific entry types.

Table 9-4 **SD-UX ACL Entry Key Values**

Entry Type	Key Content
`user`	a user name [optionally, @ *`remote-host`*]
`group`	a group name [optionally, @ *`remote-host`*]
`host`	a host name

Table 9-4 SD-UX ACL Entry Key Values (Continued)

Entry Type	Key Content
`other`	[optionally, @ *`remote-host`*]
`any_other`	no key allowed

When listing the ACL, the remote-host is printed in its Internet address form (e.g., 15.12.89.10) if the local system cannot resolve the address from its host lookup mechanism (DNS, NIS, or `/etc/hosts`). The remote-host must be recognized (resolvable) when used in the `-M` and `-D` options. Unrecognized remote-host values are accepted in files provided with the `-F` option.

ACL Permissions

There are five different permissions grantable by the ACL: `crwit`.

Table 9-5 ACL Permissions

control (`c`)	Permission to edit or change the ACL.
test (`t`)	Permission to test access to an object (i.e., read the ACL).
insert (`i`)	Permission to install a new product, depot or root.
write (`w`)	Permission to change a host, depot, root or product.
read (`r`)	Permission to list depot, roots and products and attributes.

In the ACL entry, these permissions are abbreviated `c`, `t`, `i`, `w`, and `r`. To grant all permissions, you may use the shorthand letter `a` instead of the `crwit` to denote *all* permissions.

The meaning of permissions is different for different types of objects, and the permissions do not have to appear in any specific order. Roots do *not* provide product level protection, so all permissions on products installed on roots are controlled by the ACL protecting the root itself.

Product level protection is provided on depots in this way: the depot's ACL protects the depot itself while product ACLs protect the products within the depot.

The table below summarizes SD-UX object permissions and ACLs to which they may be applied.

Table 9-6 SD-UX ACL Permission Definitions

Permission	**Allows You To:**			
	Host System	*Root*	*Depot*	*Product on Depot*
`c` (control)	Edit all ACLs			
`t` (test)	Test access to an object, read (list) the ACL itself			
`i` (insert)	Insert a new depot or root	Insert a new product	Insert a new product	N/A
`w` (write) [a]	Change host	Change root or products	Change depot	Change product
`r` (read) [b]	List depots and roots	List root and product attributes	List depot and product attributes	Read product files

a. Write permission means permission to change or delete the object, except the host source object may not be deleted.
b. Read permission on containers (i.e., hosts, roots, and depots) lets a user list the container contents; on products within depots, read permission lets a user copy or install the product.

Object Protection

The control of product insert and delete permissions differs between roots and depots.

The permission for anyone to insert or delete a product on a root is contained within the root's ACL. If you have write permission on a root, you can change or delete any product on that root; there is NO product level control on roots.

The depot ACL controls insertion (creation) of new products, while the inserted object has its own ACL that controls modification and deletion. This lets the creator (owner) of a product on a depot change or delete the product without requiring the broader write permission that could affect other users' products on the same depot.

This is useful for product control, because it lets you assign management control for a specific product to a delegated administrator. Also, when a product is created on a depot, the user and group identity of the creator is recorded in the product information.

If the product ACL contains an `object_owner` entry granting write permissions to the owner, then the product creator will automatically have rights to change or delete the product. Therefore, the depot can be more widely opened to insertion because users with insert permission can only copy in new products or delete their own products: you don't have to worry about a user erroneously deleting some critical product that they shouldn't control.

The rationale for this protection scheme is borrowed from a mechanism introduced in the BSD file system. With `write` permissions on a BSD directory, you may create a file in the directory. If the sticky mode bit is set on the directory, only the file owner, the directory owner, or superuser may remove or rename the file.

For example: In `/tmp`, owned by root, with "wide-open" write permission and the sticky bit set manually (i.e., mode 1777), anyone can create files that nobody else (except themselves and superuser) can remove. This makes `/tmp` a more secure place to store temporary work because someone else can't delete your files there.

Installing or copying from an unregistered depot requires the user *and* the target agent's host to have insert permission on the depot's host. If this permission is denied to the target's host, the depot's daemon log will contain the message:

```
ERROR: Access denied to SD agent at host lucille on
behalf of rob@lucille to start agent on unregistered
depot "/users/rob/depot." No (i)nsert permission on
host.
  07/23/01 15:51:06 MDT
```

This message indicates it is the agent at `lucille` that did not have insert permission on the depot's host, not the user `rob@lucille`.

The remote host ACL must have two entries granting insert permission: one for the user, and one for the target host.

For example, for user `rob` to be allowed to install a product on target host `lucille` from an unregistered depot on source host `desi`, the command

```
swacl -l host @ desi
```

must show the minimum ACL entries

```
user:rob@lucille:-i-
host:lucille:-i-
```

Rob could alternatively register the depot with the swreg command with only the first entry above before running swinstall or swcopy.

Host System ACLs

The *host system* is the highest level of protected object in SD-UX. A *host ACL* protects each host system, controlling permission to create depots and roots. The host ACL may grant the following permissions:

Table 9-7 Host ACL Permissions

`r` (read)	Permission to obtain host attributes, including a list of depots and roots on the host.
`w` (write)	Permission to change the host object.
`i` (insert)	Permission to create and register a new depot or root on the host.
`c` (control)	Permission to edit or change the ACL.
`t` (test)	Permission to test access to an object and list the ACL.

A sample host-system ACL grants depot and root source creation, source listing, and ACL administration to a user named `rob` and give open permission to list the depots and roots on the host, would be:

```
user:rob:r-ic-
any_other:r
```

Since `any_other` does not have `t` (test) permission, only *`rob`* can list this ACL, because he has `c` (control permission).

Root ACLs

Principals (users) identified in ACLs that are protecting roots are granted permission to manage installed products. The permissions associated with a root are:

Table 9-8 Root Permissions

`i` (insert)	Permission to install a new product.
`r` (read)	Permission to list the contents of the root.
`w` (write)	Permission to delete the root itself or the products in the root.
`c` (control)	Permission to edit or change the ACL.
`t` (test)	Permission to test access to an object and list the ACL.

A sample root ACL that grants a user named `lois` permission to read, write, and insert software and members of the group named `swadm` all possible permissions is:

```
user:lois:rwi-
group:swadm:crwit
```

When a root is created, it is automatically protected by a default ACL derived from its host. Use swacl to change the initial values of this ACL. For additional information, see “ACL Templates” on page 282.

Depot ACLs

Principals identified in ACLs that are protecting depots are users who have been granted permission to manage the depot and to create new products. The permissions associated with a depot are:

Table 9-9 Depot Permissions

`i` (insert)	Permission to copy a new product into the depot.
`r (read)`	Permission to list the contents (products) of the depot source.
`w (write)`	Permission to delete the depot (if it is empty), and unregister itself (not the products in the depot).
`c (control)`	Permission to edit or change the ACL.

Table 9-9 Depot Permissions (Continued)

`t (test)`	Permission to test access to an object and list the ACL.

A sample depot ACL that grants its creator all permissions; user `george` permission to list and insert software products; members of group `swadm` permission to list and insert products, change the ACL and delete the depot itself; and everyone else permission to list the contents of the depot, would be:

```
object_owner:crwit
user:george:-r-i-
group:swadm:crwi-
any_other:-r-
```

When a depot source object is created, it is automatically protected by a default ACL derived from its host. Products inserted in that depot will automatically be protected by an ACL derived from the depot. This concept is discussed in the “ACL Templates” on page 282.

Product ACLs

Product ACLs only apply to products on depots. Products on roots are protected by the root’s ACL. There are two classes of principals that are granted access rights to products:

Table 9-10 Product Principals

users	Granted various administrative permissions. This class includes groups and others, both local and remote.
hosts	Target systems (agent/daemons) granted read permissions to allow product installation.

Permissions on products are:

Table 9-11 Product Permissions

`w` (write)	Permission to users to change and delete the product and/or product information.

Table 9-11 Product Permissions (Continued)

`r` (read)	Permission granted to target_hosts to read the source-depot product. (that is, grant permission to a remote system to install the protected product).
`c` (control)	Permission to edit or change the ACL.
`t` (test)	Permission to test access to an object.

A sample product ACL that grants user *`swadm`* and the creator of the product all permissions and allows open read permission (allowing free distribution to all systems) would be:

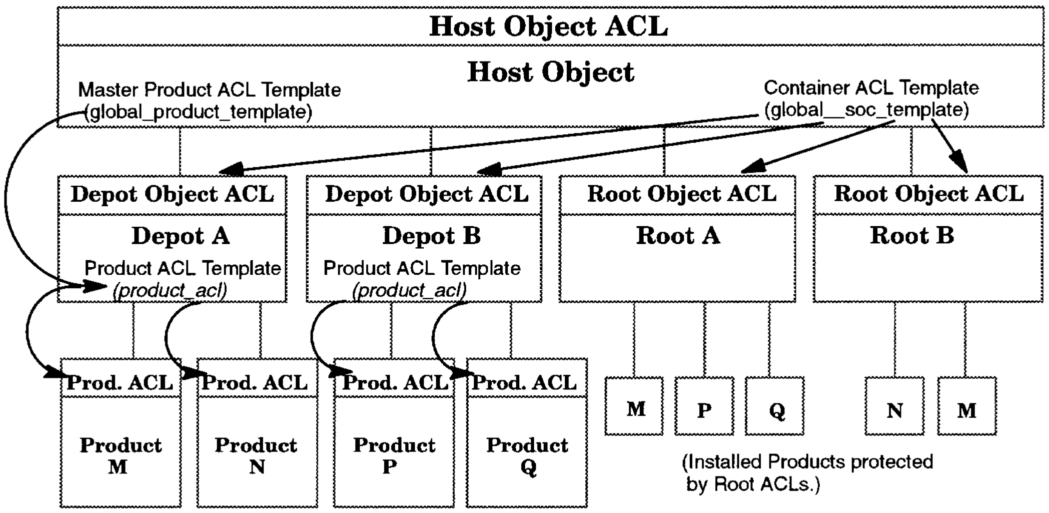
```
user:swadm:crw
object_owner:crw
any_other:-r-
```

NOTE When a product object is created, it is automatically protected by a default ACL from the depot/root source or, absent that, one from the host.

ACL Templates

There are two ACLs that are used to create the initial ACLs that protect newly created objects: **product ACL templates** (`global_product_template` or `product_template`) and **container ACL templates** (`global_soc_template`).

Figure 9-2 ACL Templates



When a product is put into a depot with swcopy or swpackage, SD-UX uses a product ACL template (provided by the depot that contains that product) to define the initial permissions of the new product's ACL.

SD-UX uses the product ACL template of the host system (`global_product_template`) to initialize the product ACL template of the new depot and uses the container ACL template of the host system (`global_soc_template`) to initialize depot and root ACLs.

Thus, there are three ACLs on the host:

- Host ACL

  Attached to and controlling access to the host object itself.

- Container ACL Template (`global_soc_template`)

  Used to initialize the ACL protecting new depots and roots created on the host.

- Product ACL Template (`global_product_template`)

  The ACL that is used to initialize the product ACL template on depots that are created on the host.

There are also two ACLs on product depots:

- The depot's ACL that is used to determine permissions on the depot.
- The depot's product ACL template (`product_template`) that is used to initialize the ACLs protecting new products on the depot.

There is one ACL on the installation (root):

- The root ACL that protects the root and products installed on it.

And finally, there is one ACL on the product:

- The product's ACL that is used to determine permissions on the product.

Every host must have an ACL protecting it and a pair of template ACLs (product and container) to provide initialization data for implicit depot and product ACLs. All three are created when SD-UX is installed on the host.

Default ACL Template Entries

The host system's container ACL template dictates initial permissions on all depots and roots that are introduced on that host. The host also contains a master copy of a product ACL template, which is copied to each new depot.

A default set of host ACLs is provided at the time SD-UX is installed that can be altered by the SD-UX administrator. The contents of these host-system ACLs immediately after SD-UX installation are:

Host ACL

- The host ACL below allows global (`any_other`) permission to list the depots and roots on the host:

  ```
  object_owner:swadm:crwit
  any_other:-r---
  ```

NOTE Remember, the local superuser always has all permissions, even without an ACL entry.

Container ACL Template

- The container ACL template below grants the owner or creator (`object_owner`) of a new depot or root permission to manage that new depot or root and to change its ACL. It also grants global permission (`any_other`) to list products in the new depot or root.

  ```
  object_owner:crwit
  any_other:-r---
  ```

Product ACL Template

- The product ACL template below grants permission to perform all operations on products installed on Depots on this host to the respective creator (i.e., owner), via the `object_owner` entry, of each product. It also grants permission to read (i.e., install) and test the product to any host (the `any_other` entry).

  object_owner:crwit
  any_other:-r---

- In addition to encompassing all hosts, the `any_other` entry also applies to all other users except, in this case, the product's owner. In SD-UX however, product read permission has meaning only to host principals, and other possible product permissions never apply to hosts; therefore, the `any_other` entry may be overloaded with user and host permissions, if desired, without any danger of ambiguity. This overloading should be kept in mind when using the SD-UX to execute solutions.

These host ACL defaults provide a good starting point for control over the management functions of SD-UX while providing open access to read the software for installation on root targets.

Security on SD-UX Systems

Controlling access to data is a key concern of computer security. In SD-UX, file owners and superusers allow or deny access to files on a need-to-know basis by setting or manipulating the file's permission bits to grant or restrict access by owner, group and others. For example, the following file listing:

```
-rwxr-xr  1 doug   admin   738 Mar 26 12:25  datafile
```

shows that:

- File owner is user `doug.`
- File's group is `admin.`
- Name of the file is `datafile`.
- Owner permissions are `read`, `write` and `execute` (rwx).
- Group permissions are `read` and `execute` (r-x).
- Other permissions are `read only` (r-).

SD-UX commands are essentially object managers that use the SD-UX file system in which to store their objects. There is no need to obtain access to any objects via the file system, so the file system protection scheme is based on blocking access to the file system directories that store these objects.

In addition to SD-UX objects, there are several administrative files (log, configuration, and session files) that are used or managed by SD-UX. These files are not actually SD-UX objects and are accessible via conventional commands such as editors and printing utilities. These files are protected by conventional file system protection modes.

Many of the functions that the SD-UX agents do are privileged. Some operations, such as installing files in system directories (e.g., in the `/etc` and `/dev` directories) and customization of system files via control scripts, require superuser privileges. For this reason, SD-UX agents must always run as the superuser.

Any system user may run the SD-UX controller; it is not restricted to use only by superuser. In general, the controller does its work by making Remote Procedure Calls (RPC) to target hosts, but it also requires special privileges occasionally to access critical log, configuration, and session security files. Controllers are `set-uid root` programs that run with the superuser privilege in effect only briefly to do critical privileged operations, then they switch to the real `uid` of the user.

Here is a summary of the SD-UX file system protection scheme:

- SD-UX files are protected from access by anyone other than the superuser by having the group and other permissions of crucial directory modes set to 0.
- Only agents and daemons running on the local host access SD-UX files directly. All other facilities (controllers, utilities, etc.) go through the agents using RPC to indirectly access files. The agent or daemons perform authentication and authorization checks on all such operations.
- No hard links may exist that circumvent the directory protection hierarchy of the SD-UX directories nor may symlinks exist that compromise the secrecy of the contents of those directories containing objects that might have list restrictions in effect. Use of only a single (canonical) path to SD-UX objects avoids any such aliasing problems.

Thus, the SD-UX files are totally protected and hidden from non-superuser access.

SD-UX Internal Authentication

This section discusses the following topics:

- SD-UX Credentials
  - Controllers Run with the User’s Credentials and Privileges
  - Agents Run with the System’s Identity
- Security Between Hosts: The Shared Secrets File

SD-UX security does not replace DCE Security. It seeks to provide a usable protection scheme based on the assumption that there is no hostile, concerted effort by users to do damage.

Much of the DCE security functionality used by SD-UX comes from the DCE Runtime Library that is included in SD-UX. This library provides DCE RPC capability and some of the DCE Security Services required to support ACLs.

Without full DCE Security Services, it is impossible to reliably prove the identity of a user making an SD-UX RPC call; even if the source and destination of the RPC call is local. The RPC identifies only the network address of the calling client.

This means that a person who has access to a legitimate SD-UX host system and knows the SD-UX call interface and protocol could impersonate an SD-UX controller. This would create a significant security risk in a hostile environment.

However, SD-UX makes it possible to run securely without these DCE Security Services by providing its own internal method of performing user, group, and host authentication.

SD-UX Credentials

A key to SD-UX security is determining which users are allowed to be involved in particular operations. In SD-UX internal authentication, your HP-UX `uid`, `gid`, and host name are used to establish your identity. The fact that the SD-UX controller runs with an effective `uid` of root (because the controller is a `setuid-root` program) does not affect your identity, which is obtained from your real `uid.`

When you start an RPC (as an SD-UX controller), a structure describing your identity accompanies each call to an agent; the controller sends the user and group name of the person invoking the RPC, as well as the host name of the system on which it is running (in DCE, called the **realm**).

This structure is called your **credentials**. Credentials consist of:

- **user (principal) name**

  The user (or host system, for agents making RPCs to other agents) who is originating the RPC call.

- **Group name**

  The user's primary group.

- **Realm or local Host**

  The user's host name.

The user's credentials are passed in the RPC parameters, The agent receiving the RPC uses this information to compare authentication credentials.

Controllers Run with the User's Credentials and Privileges

SD-UX controller programs such as swinstall or swremove operate with the privileges of the user who invokes them. The agent ensures that the user has the required permissions on the object by looking at the object's ACL. If permissions are not granted, the operation fails.

A controller may be run by anyone on the system, but its actions are restricted (based on permissions granted in various object ACLs). SD-UX agents always verify that user-requested operations are authorized before performing them.

Agents Run with the System's Identity

The SD-UX agents and daemons run with the privileges of a superuser; but they also have the special identity of the host system on which they are executing. When a *target* agent makes an RPC call to a *source* agent, two sets of credentials are passed with the call:

- those of the agent's system
- those of the user running the controller on whose behalf the target agent runs

While local superuser privilege is necessary for the agent to do required local file system operations such as file creation and deletion, ACL management, etc., this level of permission is neither required nor desired for DCE RPC operations with other SD-UX processes.

When SD-UX agents perform RPCs, they assume the identity of the system on which they run, rather than that of a particular user.

Security Between Hosts: The Shared Secrets File

In addition to the caller's credentials, another proof of trustworthiness is also sent in the RPC. The SD-UX agent checks this proof before accepting the caller credentials. This proof consists of passing the encryption of a secret password. The password is read from the shared secrets file. This file is located on systems in `/var/adm/sw/security/secrets`.

NOTE The SD-UX Secret must be the same on both the target system and the controller.

The agent compares this encrypted secret to the encryption of a local secret it shares with the controller's host. If the secrets do not match, the call is not authenticated and it fails.

Secrets are stored by host name in the secrets file and are used to establish trust between two systems. The controller selects a secret in the file that corresponds with the host name of the system on which it is running. The agent, upon receipt of an RPC from the controller, looks up a secret associated with the controller's host.

For example, if the *controller* is running on `alma.fc.hp.com` and makes a request of an *agent* running on `lehi.fc.hp.com,` each of the two processes will look up the secret associated with `alma.fc.hp.com` (the controller's host) from their respective secrets file.

Here is an example of the format of the shared secrets file:

```
default             quicksilver
lehi.fc.hp.com      s28ckjd9
alma.fc.hp.com      32hwt
newdist.fc.hp.com   zztop
noway.fc.hp.com     daisey
```

The first column represents the controller's host name and the second column represents the controller's secret.

There is also a provision for a default secret (`quicksilver` in the example above), to be used when no system name match is found in the secrets file. The entry is identified with the default pseudo-host name. This entry allows open SD-UX interconnect between hosts sharing the same default entry. SD-UX is shipped with the secret `-sdu-` that should be changed for your site.

When you change a host's secret, make sure you change it in the secrets files of all hosts with which you work. The secrets file may be produced in a single site, then copies distributed to all participating hosts.

NOTE The secrets discussed here does not grant any access to SD-UX objects, but do allow a host to participate in SD-UX operations.

RPC Authorization

This section discusses how agents handle controller requests, local superuser authorization, depot registration, and daemon/agent security

In SD-UX, objects are protected by ACLs. An ACL is a structure, attached to an object, that defines access permissions for multiple users and groups. It extends the concepts defined by the HP-UX file system mode bits in two ways: by allowing specification of the access rights of many individuals and groups instead of just one of each; and by protecting entire SD-UX objects, rather than individual files.

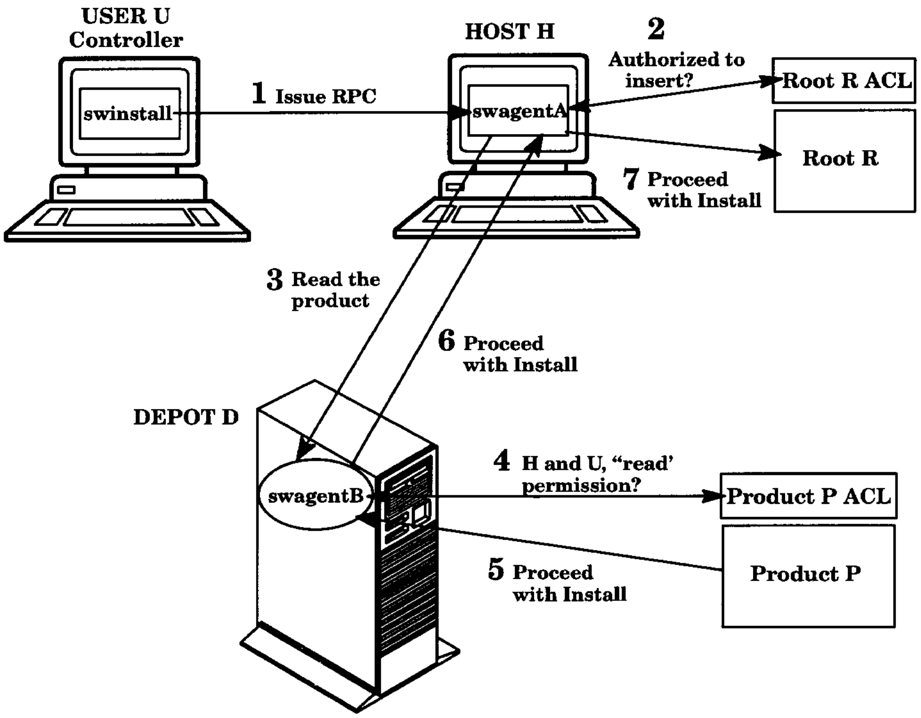
Generally, a controller requests an agent to perform some operation on a object. SD-UX protects each host, depot, depot-product, and installation object (root) with an ACL. After a call is authenticated, the ACL manager is consulted for a caller's access permissions to a protected object before allowing the action.

SD-UX authorization uses ACLs to determine the RPC caller's rights to access a particular SD-UX object in a particular way (i.e., read, write). An object's ACL is searched for an entry that matches the caller. Once a matching entry is found, the permissions granted in that entry are compared to those required for the operation. If permissions required for the operation are all granted by the entry, access is authorized, and SD-UX proceeds with the requested operation.

How Agents Handle Controller Requests

When a controller requests an agent to do an operation requiring the participation of another agent, the two agents must each grant access to the objects under their control before the operation can complete.

Figure 9-3 SD-UX Security Process



For example, to install a product P from depot D to root R:

1. User U sends an RPC request to swagentA on the target host H. User U wants to install the product in root R (on the target host).
2. SwagentA checks the ACL protecting root R to confirm that user U is authorized to insert products.
3. SwagentA (running as principal H) forms a request to swagentB (running where depot D resides) to read the product.

4. SwagentB checks the ACL protecting the product to make sure that both the destination system (principal H) and the user U have read permission before honoring the request, and the installation proceeds.

The ACL on swagentB neither knows of nor depends on user U. The ACL on root R acts to screen U; then (and only then) the product's ACL acts to screen H.

As a special case, the superuser always has full permissions on a local system.

Local Superuser Authorization

As a special case, SD-UX always allows the local superuser full access to all local objects regardless of ACL protections. This allows the local superuser to repair corrupted ACLs or to perform any other operations.

Delegation

SD-UX provides a form of **delegation** to control access to depot-resident products: both the host where the target agent is running and the user initiating the call must have read access.

This form of delegation passes the caller credential information to the depot agent in the RPC options. This form of delegation works the same whether the agents are configured to use DCE or SD-UX Internal authentication.

It is important to note that this delegation technique is provided to allow user-level access to depot-resident products.

Depot Registration and Daemon/Agent Security

Because SD-UX stores its objects in the file system, someone could build a "Trojan Horse" file system image of a software depot. This could breech the security of any system that installed products from the false depot. To protect systems from such a situation, SD-UX requires that a depot be registered with SD-UX (either through swcopy or by using swreg) before software may be installed or copied from it. This check is always performed before granting access. Registration with swreg requires *insert* permission in the host's ACL.

As a special case, an unregistered depot may be used for local installation (i.e., the depot and destination root exist on the same system) if the initiator is the local superuser or has permission to register the depot (*insert* permission on the host).

The administrator of a host system must ensure the integrity of new depots before registering them and ensure that only trustworthy users are granted permission to insert on the host.

NOTE In addition to registering users, caution should be exercised when installing or copying from unregistered depots.

Security Use Models

The use models below use the `swadm` group that is provided in the default host ACLs, which are installed at SD-UX install-time. This group is not a part of the default HP-UX configuration, but can be easily added. First, add the `swadm` group and the appropriate group members by using the HP-UX System Administration Manager product. Next, provide the `/etc/logingroup` link to `/etc/group` to activate HP-UX supplementary groups.

NOTE `/etc/logingroup` is an HP-UX utility to support both SVR2/3 and BSD group semantics selectively. When `/etc/logingroup` is linked to `/etc/group`, HP-UX gives BSD (and SVR4) semantics.

If the file `/etc/logingroup` does not exist on systems targeted as SD-UX Controllers, execute the following command (as superuser) on each appropriate system:

```
ln -s /etc/group /etc/logingroup
```

Security in Remote Distributions

A common use of SD-UX remote operations capabilities is for a software administrator to **push** software from a local depot out to numerous remote targets.

You can set up of this kind of configuration:

1. Establish the group `swadm` on the controller host as described above.
2. Edit the three host ACLs on each target system. If you used the suggested setup discussed in “Setting Up Remote Operations” on page 199 to install the agents on the target systems, you may edit the three host ACLs on the Targets as superuser on the system from which you performed setup:

```
swacl -l host \
   -M group:swadm@`hostname`:a @ remsys1. . .remsysN
swacl -l global_soc_template\
   -M group:swadm@`hostname`:a @ remsys1. . .remsysN
swacl -l global_product_template \
   -M group:swadm@`hostname`:a @ remsys1. . .remsysN
```

You may want to grant permissions to specific users to manage particular products on the primary depot. For example, user `ramon` may be assigned responsibility to manage the ALLBASE product on your depot, installing new versions and patches when they become available. To add `ramon` to the ACL for ALLBASE on the local depot and grant him all permissions on that one product, run the command:

```
swacl -l product -M user:ramon:a ALLBASE
```

At the same time, you may want to eliminate the ACL entry for group `swadm` for the same product:

```
swacl -l product -D group:swadm ALLBASE
```

Security in Local Distributions

Host administrators may grant permission to individual users or groups, trusted at the local host, to administer software locally. Trusted local users have root ACL entries granting insert and write permissions. At the source depot, access to all software products is allowed by unrestricted read access to hosts, depots, and products. This is the basis of a pull model of software distribution.

Restricting Installation to Specific Target Systems by Specific Users

Managers of software source depots may leave software openly installable, as described above, or may choose to limit distribution to specific systems. ACLs protecting source depot products may contain entries that restrict product read access to only specified systems, allowing installation only to those systems. This restriction applies to both the push and pull models.

Below is a sample product ACL that restricts read permission to systemA and systemB and grants all permissions to user `swadm`:

```
user:swadm:rwict
host:systemA.loc.company.com:r
host:systemB.fc.hp.com:r
```

Security for Software Developers

Software developers iteratively package their products and test them before distribution. This involves packaging products into depots and installing them to Roots for testing. Since it may require several iterations to get all the customization right, it is not helpful to prevent software developers from having free access to depots and Roots for this testing.

You should also not have products that are being tested, coming and going on wide-use depots and roots. They might accidentally be installed or used before they are ready.

The recommended method of development is to provide one or more development depots and roots for testing purposes, each with protections customized to meet the needs of the development group using them. To this end, the default ACL template mechanism described previously is handy, since products come and go quickly.

A host administrator (someone with insert permission on the host) should create the test depot for developers, then assign a depot administrator and edit the depot ACL to grant that person control (ACL edit) permission on the depot. The depot's product ACL template should then be set up so that users inserting a product may also write (modify and delete) it, and so that it may be read only by the known test systems.

Similarly, test roots may be created, perhaps on other test hosts, to which developers may install test products. Access to install to the test root should be restricted to the development group.

When testing is complete and a product is ready for release, the product may then be copied to a general distribution depot to make it more widely readable without exposing all the untested products on the test depot.

There are many additional ways in which these basic concepts may be used to implement a desired security policy for product development.

Permission Requirements, by Command

Packaging (swpackage)

- If the depot does not exist, swpackage verifies that the user has insert permission on the target host.
- swpackage verifies that the user has insert permission on a target depot.
- swpackage verifies that the user has write permission on target product, if it already exists.

Listing (swlist)

- To list potential depots, the source agent verifies that the controller user has read permission on host.
- To list potential products, the source agent verifies that the controller user has read permission on depot or root.

Job Browsing (sd, swjob)

- To use the CLI (swjob) or GUI (sd) to view information about jobs initiated from a local host, the controller verifies that the user has read permission on the host.
- To use the command line or GUI to retrieve a target log file, the target agent verifies that the controller user has read access on the root or depot target.

Copying (swcopy)

- Any list operations required to facilitate this function must be checked as described in the swlist section above.
- If the depot does not exist, swcopy verifies that the user has insert permission on the target host.
- The target agent verifies that the controller user has insert permission on the target depot.

- The target agent verifies that the controller user has write permission on the target product, if it already exists.
- The source agent verifies that the target agent system has read permission on the source product.
- The source (depot) agent verifies that the depot is registered. If not, the agent verifies that the controller user and the target agent system each has insert permission on the source's host.

Installing (swinstall)

- Any list operations required to facilitate this function must be checked as described in the swlist section above.
- The target agent verifies that the controller user has insert permission on the target root.
- The target agent verifies that the controller user has write permission on the target root, if the product already exists.
- The source (depot) agent verifies that the target agent system has read permission on the source product.

Removal (swremove)

- If the object is a product on a depot, the target agent verifies that the controller user has write permission on the target product.
- If the object is a product on a root, the target agent verifies that the controller user has write permission on the target root.
- If the object is a depot or root, or the last product contained in one of these, before removing the container the target agent must verify that the controller user has delete permission on the target root or depot.

Configuration (swconfig)

- The same permission checks are made as for the swremove operation above, except that this command does not apply to depots.

Verify (swverify)

- If the object is a product on a depot, the target agent verifies that the controller user has read permission on the target product.
- If the object is a product on a root, the target agent verifies that the controller user has write permission on the target root (since scripts are executed).

Registering Depots (swreg)

- To register a new depot, the target daemon verifies read permission on the depot to be registered and insert permission on the host.

Changing ACLs (swacl)

- To change an ACL, write permission is required.
- To list an ACL, list permission is required.

Request Scripts (swask)

- To query a user and obtain installation information, interactive control scripts are used.

Modify (swmodify)

- To change or add information to the Installed Products Database (IPD) or depot catalog files, write permission is required.

10 Creating Software Packages

This chapter describes the tasks associated with packaging software for distribution.

Table 10-1 Chapter Topics

Topics:
"Overview of the Packaging Process" on page 302
"Identifying the Products to Package" on page 303
"Adding Control Scripts" on page 305
"Creating a Product Specification File (PSF)" on page 307
"Packaging the Software (swpackage)" on page 345
"Packaging Tasks and Examples" on page 356

Overview of the Packaging Process

To help you distribute software from depots, Software Distributor lets you package software into SD-UX format. The packaging process lets you create depots directly or create packages that you can add to depots later. The packaging specification is flexible enough to fit many software build and manufacturing process needs.

The packaging process consists of the following tasks:

1. Identifying the package.

   Determine what files and directories you want to include in your software package, and determine product structure. Your software package can consist of files, filesets, subproducts, products, and bundles.

2. Write control scripts (optional).

   You can write control scripts and include them in your package. These scripts let you perform additional checks and operations beyond those supported by SD-UX.

3. Create a **Product Specification File** (**PSF**) to define the product package.

4. Create the software package by running the swpackage command.

   The swpackage command reads the PSF file, analyzes the product definitions, and packages the source files and information into product objects. It then creates and inserts the product into the distribution depot.

Prerequisites

Before you begin packaging software, ensure the following:

- SD-UX is installed and configured on the system where you intend to create your software package.
- The software to package is installed on the packaging system, or that the necessary files are available remotely.

Identifying the Products to Package

Determining Product Contents

The first step in packaging software is to determine what files and directories you want included in the software product. These files and directories must follow certain guidelines to support the configuration you want.

Key points in this structure are:

- Where are shareable (for example, executables) and non-shareable (for example, configuration) files installed?
- How is configuration used to put non-shareable files in place?

Determining Product Structure

Determine the product structure that your software should follow. SD-UX provides four levels of software objects:

Level	Objects
Filesets	(Required) Filesets include the actual product files, information that describes those files (attributes) and separate control scripts that are run before, during or after the fileset is installed, copied or removed. Filesets are the smallest manageable (selectable) software object. Files must be grouped into one or more filesets. Filesets must be grouped into one or more products. (Filesets can be members of only a single product.)
Subproducts	(Optional) Subproducts are used to group related filesets within a product if the product contains several filesets. Subproduct definitions are optional.
Products	(Required) Filesets (and/or subproducts) must be grouped into one or more products. They are usually grouped into collections that form a set of related software, or match the products that a customer

	purchases. The SD-UX commands maintain a product focus, while still allowing the flexibility to manage subsets of the products via subproducts and filesets.
Bundles	(Optional) Bundles are provided only by the HP factory. Customer packaging of bundles is not supported.

NOTE You can define different versions of products for different platforms and operating systems, as well as different revisions (releases) of the product itself. You can include different product versions on the same distribution media.

Adding Control Scripts

SD-UX supports execution of product and fileset control scripts that allow you to perform additional checks and operations with other HP-UX commands and functions. The swask, swinstall, swconfig, swverify, and swremove commands each can execute one or more control scripts on the primary roots. You can write the scripts and include them in your software package. All scripts are optional but many times are needed correctly complete the task that you want your software package to perform. See Chapter 11, "Using Control Scripts," on page 369 for a complete discussion of control scripts.

SD-UX supports the following types of scripts, which can be defined for products and fileset:

`Checkinstall`	Analyses each target to determine if the installation and configuration can take place. (Executed by swinstall.)
`Checkremove`	Analyses each target to determine if removal and unconfiguration can take place. (Executed by swremove.)
`Configure`	Configures installed filesets or products. (Executed by swconfig and swinstall.)
`Fix`	Corrects and reports on problems in installed software. (Executed by swverify.)
`Postinstall`	Performs additional install operations (such as resetting default files) immediately after a fileset or product has been installed. (Executed by swinstall.)
`Postremove`	Performs additional remove operations (such as restoring "rollback" files) immediately after a fileset or product has been removed. (Executed by swremove.)
`Preinstall`	Performs file operations (such as removing obsolete files) immediately before installation of software files. (Executed by swinstall.)
`Preremove`	Performs additional file operations (such as removing files created by a preinstall script) immediately before removal of software files. (Executed by swremove.)

`Request`	Requests an interactive response from the user as part of the installation or configuration process. (Executed by swask, swconfig, and swinstall.)
`Unconfigure`	Undoes configurations performed by configure scripts. (Executed by swconfig and swremove.)
`Unpostinstall`	Undoes a postinstall script in case swinstall must initiate recovery during the installation process. (Executed by swinstall.)
`Unpreinstall`	An undo preinstall script in case SD must initiate recovery during the install process. (Executed by swinstall.)
`Verify`	Verifies the configuration of filesets or products in addition to the standard swverify checks. (Executed by swverify.)

Creating a Product Specification File (PSF)

SD-UX uses a Product Specification File (PSF) to define the physical product package. The PSF provides a “road map” that identifies the product according to its attributes, contents, compatibilities, dependencies and descriptions. The PSF drives the swpackage session. It describes how the product is structured and defines the attributes that apply to it.

SD-UX packages, distributes, installs files. The SD-UX packager uses these files after they have been built and installed into specific directory locations. These directory locations my reside in separate, unconnected directory trees or in the specific file locations needed to make the software run on your system. You can specify files by a root directory (gathering all files below it) or by explicit individual file paths. The file attributes can be taken from the files themselves, specified separately for each file, or specified for a set of files.

The PSF can:

- Define vendor information (optional) for groups of products (including all products), or for individual products.
- Specify one or more products (required).
- For each product, define attributes for one or more subproducts (optional), filesets (required), and files (required).
- Define attributes for the distribution depot/media (optional).
- Specify what computer(s) and operating system(s) the product supports.
- Define attributes that describe the software objects.

Product Specification File Examples

Minimal PSF

Here is an example of the minimum PSF, which includes only the required keywords. This PSF creates a product `SD` with fileset `commands` and contains one file, `/usr/sbin/swcopy`:

```
product
   tag   SD
fileset
  tag   commands
file    swcopy /usr/sbin/swcopy
```

NOTE You must use an absolute path for the second file term in this minimum format.

Typical PSF

Here is a sample PSF that describes the SD-UX product:

```
# PSF defining SD as a sample product.
depot
  layout_version 1.0
# Vendor definition:
vendor
  tag          HP
  title        Hewlett-Packard Company
  description  < data/description.hp
category
  tag           system_mgt
  title         Systems Management Applications
  description   These are the system management
applications
  revision      1.0
end
# Product definition:
product
  tag           SD
  revision      A.01.00
  architecture  HP-UX_B.11_32/64
  vendor_tag     HP
  is_patch       false
  title          HP-UX Distributor
  number         B2000A
  category_tag   system_mgt
  description    < data/descr.sd
  copyright      < data/copyr.sd
  readme         < data/README.sd
  machine_type   *
  os_name        HP-UX
  os_release     ?.11.*
  os_version     ?
  directory      /
  is_locatable   false
# Specify a checkremove script that executes during the
# swremove analysis phase. (This script prevents the
# removal of the SD product and returns an ERROR.
    checkremove     scripts/checkremove.sd

# Subproduct definitions:
  subproduct
    tag          Manager
    title        Management Utilities
    contents     commands agent data man
  end
  subproduct
    tag          Agent
    title        Agent component
    contents     agent data man
  end
 # Fileset definitions:
  fileset
    tag          commands
    title        Commands (management utilities)
    revision     2.42
    description  < data/descr.commands
 # Dependencies
    corequisites  SD.data
    corequisites SD.agent
 # Control files:
    configure    scripts/configure.commands
 # Files:
    directory    ./commands=/usr/sbin
    file         swinstall
    file         swcopy
# (...Other file definitions can go here...)
    directory    ./nls=/usr/lib/nls/C
    file         swinstall.cat
    file         swpackage.cat

    directory    ./ui=/var/adm/sw/ui
    file         *
# (...Other file definitions can go here...)
 end
 # Commands
 # (...Other fileset definitions can go here...)
 # Manpage fileset definitions:
 fileset
    tag          man
    title        Manual pages for the SD-UX
    revision     2.05
    directory    ./man/man1m=/usr/man/man1m.Z
    file         *
    directory    ./man/man4=/usr/man/man4.Z
    file         *
    directory    ./man/man5=/usr/man/man5.Z
    file         *
  end
 #man
end
#SD
```

PSF Syntax

Each SD-UX object (product, subproduct, filesets, and file) has its own set of **attributes** and each attribute has a **keyword** that defines it. Most attributes are optional; they do not all need to be specified in the PSF. Each attribute has its own specific requirements, but the following rules apply:

- Keyword syntax is:

  ```
  keyword value
  ```

- All keywords require one or more values, except as noted. If the keyword is there but the value is missing, a warning message is generated and the keyword is ignored.
- Place comments on a line by themselves or after the keyword-`value` syntax. Comment lines are designated by preceding them with #.
- Use quotes when defining a value (for example, description) that can span multiple lines. Quotes are not required when defining a single-line value that contains embedded whitespace.
- Any errors encountered while reading the PSF cause swpackage to terminate. Errors are also logged to both `stderr` and the logfile.

PSF Object Syntax

The following tables and sections describe the PSF keywords, the allowable values for each keyword, and the syntax for the objects you can define in a PSF.

- Keywords marked with a + apply to products only.
- Keywords marked with a `-` apply to bundles only.
- Keywords marked with a * are of the `version_component` type, as well as the type indicated in the table.

Table 10-2 Keywords Used in the Product Specification File

Keyword	Value	Max. Size in bytes	Example
Distribution Class			

`distribution`
`layout_version`
`tag`
`copyright`
`description`
`number`
`title`
`end`
	
	`revision_string`
	`tag_string`
	`multi_line_string`
	`multi_line_string`
	`one_line_string`
	`one_line_string`
		
		64
		64
		8K
		8K
		256
		256
			
			`1.0`
			`EXAMPLE_DEPOT <`
			`data/copyr.depot`
			`data/descr.depot`
			`B2358-13601`
			`Example packages`

Vendor Class			

`vendor`
`tag`
`description`
`title`
`end`
	
	`tag_string`
	`multi_line_string`
	`one_line_string`
		
		64
		8K
		256
			
			`HP`
			`<data/desc.hp`
			`HP Company`

Category Class			

`category`
`tag`
`description`
`revision`
`title`
`end`
	
	`tag_string`
	`multi_line_string`
	`revision_string`
	`one_line_string`
		
		64
		8K
		64
		256
			
			`patch_normal`
			`Normal problems`
			`0.0`
			`Category of Patches`

Table 10-2 Keywords Used in the Product Specification File (Continued)

Keyword	Value	Max. Size in bytes	Example
Product Class			
product or bundle			
* tag	tag_string	64	SD-UX
* architecture	one_line_string	256	HP-UX_B.11.11_32/64
category_tag	one_line_string	256	Systems Management
- contents	repeatable list of	none	pr.fs,r=1.0,a=,v=
	software specs		
copyright	multi_line_string	8K	<data/copyr.sd
description	multi_line_string	8K	<data/descr.sd
directory	path_string	255/102	/
is_locatable	boolean	4	false
is_patch	boolean	9	false
machine_type	uname_string	9	9000/800
number	one_line_string	64	B2000A
os_name	uname_string	256	HP-UX
os_release	uname_string	64	?.11.*
os_version	uname_string	64	A
+ postkernel	path_string	64	/usr/bin/kern_bld
+ readme	multi_line_string	255/102	<data/README.sd
+ revision	revision_string	4	A.01.00
+ share_link	one_line_string	8K	
title	one_line_string	64	Software Distributor
* vendor_tag	tag_string	256	HP
end		256	
		64	
Subproduct Class			
subproduct			
tag	tag_string	64	Manager
contents	one-line list of	none	commands agent data man
	tag string values		
description	multi_line_string	8K	<data/desc.mgr
title	one_line_string	256	Management Utilities
end			

Table 10-2 Keywords Used in the Product Specification File (Continued)

Keyword	Value	Max. Size in bytes	Example
Fileset Class			
fileset			
* tag	tag_string	64	commands
ancestor	repeatable list of	none	prod.oldfileset
	product. fileset		oldprod.fileset
architecture	revision_string	64	HP-UX_B.11.11_32/64
category_tag	tag_string	64	patch_normal
corequisite	software_spec	none	SD-UX.man.r>=2.0
description	multi_line_string	8K	<data/descr.cmd
exrequisite	software_spec	none	SD-UX.data,R>=2.1
is_kernel	boolean	9	false
is_patch	boolean	9	false
is_reboot	boolean	9	false
is_sparse	boolean	9	false-
machine_type	uname_string	64	9000/8*
os_name	uname_string	64	HP-UX
os_release	uname_string	64	?.11.*
os_version	uname_string	64	A
prerequisite	software_spec	none	SD-UX.agent,r>=2.0
* revision	revision_string	64	2.42
supersedes	software_spec	none	product.fileset,
title	one_line_string	256	fr=revision
end			SD-UX Commands
control_files Class			
control_files			
directory	path_mapping_string	none	./commands=/usr/sbin
file_permissions	permission_string	none	-u 0222 -o root -g sys
file	file_specification	none	-m 04555 bin/swinstall
end			(or) *

Table 10-3 Control File Attributes

Keyword	Type	Size in Bytes	Example
`checkinstall`	`path_string`	`1K`	`./scripts/checkinstall`
`checkremove`	`path_string`	`1K`	`./scripts/checkremove`
`configure`	`path_string`	`1K`	`./scripts/configure`
`control_file`	`path_string`	`1K`	`./scripts/subscripts`
`fix`	`path_string`	`1K`	`./scripts/fix`
`postinstall`	`path_string`	`1K`	`./scripts/postinstall`
`postremove`	`path_string`	`1K`	`./scripts/postremove`
`preinstall`	`path_string`	`1K`	`./scripts/preinstall`
`preremove`	`path_string`	`1K`	`./scripts/preremove`
`request`	`path_string`	`1K`	`./scripts/request`
`unconfigure`	`path_string`	`1K`	`./scripts/unconfigure`
`unpreinstall`	`path_string`	`1K`	`./scripts/unpreinstall`
`unpostinstall`	`path_string`	`1K`	`./scripts/unpostinstall`
`verify`	`path_string`	`1K`	`./scripts/verify`

Control Files SD-UX supports execution of **control files** (also known as control scripts) at the product and fileset level. Control scripts let you perform additional checks and operations. The swinstall, swconfig, swverify, and swremove commands each execute one or more vendor supplied scripts. All scripts are optional but many times are needed correctly complete the task that you want your software package to perform. See Chapter 11, “Using Control Scripts,” on page 369 for a complete discussion of control scripts.

Selecting the PSF Layout Version

You can select the layout version in the depot definition in the PSF (see “Product Specification File Semantics” on page 321) or with the `layout_version` option for swpackage, swmodify, swcopy, or swlist.

PSF syntax conforms to the `layout_version=1.0` of the IEEE Standard 1387.2: Software Administration(POSIX). Previous versions of SD supported the POSIX `layout_version=0.8` syntax, which continues to be supported.

Software depots cannot mix layout versions; they must be one or the other.

Differences between the two layout versions include the following:

- The vendor specification is handled differently.

  For the current standard (`layout_version=1.0`), each vendor class definition is associated only with subsequent products or bundles that contain a `vendor_tag` attribute that matches the `tag` attribute within the vendor class definition.

  For the previous standard (`layout_version=0.8)` or if you do not specify a `layout_version`, products or bundles are automatically associated with the last vendor class you defined at the distribution level, or from a vendor that you define within the product or bundle. Explicitly defined `vendor_tag` attributes (with or without a value) take precedence.

- The corequisites and prerequisites have singular titles for `layout_version=0.8` (that is, corequisite and prerequisite). See “Dependency Specification” on page 335 for more information.

- Category objects and keywords are handled differently.

  For `layout_version=1.0` (current standard):

  — `category_tag` is a valid product attribute that replaces the `category` and `category_title` attributes.

  — You can define `category` class objects.

  For `layout_version=0.8` (previous standard:

  — `category` and `category_title` are valid product attributes that replace the `category_tag` attribute.

  — `category` class objects are not recognized.

For a more complete description of PSF requirements for `layout_version=0.8`, refer to the *swpackage.4* manual page in a previous version of HP-UX.

PSF Value Types

With the exception of vendor-defined attributes (see “Vendor-Defined Attributes” on page 321), the values for each attribute keyword in your PSF must match one of the specific types discussed below.

NOTE PSF syntax conforms to the `layout_version=1.0` of the POSIX 1387.2 Software Administration standard. Previous versions of SD-UX supported the POSIX `layout_version=0.8` syntax, which continues to be supported. See “Selecting the PSF Layout Version” on page 315 for more information.

`boolean`

- Maximum length: 9 bytes
- One of the values true or false.
- Examples: true, false

`file_specification`

- Maximum length: none
- Explicitly specifies a file or directory to be packaged, using the format:

  [`-m` *`mode`*] [`-o` [*`owner`*[,]] [*`uid`*]] [`-g` [*`group`*[,]][*`gid`*]] [`-v`][*`source`*] [*`destination`*]
- The source and destination can be paths relative to source and destination directories specified in the `path_mapping_string`.
- You can also use `*` to include all files below the source directory specified by a directory keyword.
- Examples: `-m 04555 sbin/swinstall` or `*` (to denote all files and directories)

`multi_line_string`

- Maximum length: 8 kbyte (1Mbyte for a *readme* file)

- Each multi-line strings support all `isascii` characters. (Refer to the *ctype*(3) manpage.) It represent one or more paragraphs of text. It can be specified in-line, surrounded by double-quotes or read from a files.

  File entries must use this syntax:

  `< filename`
- Example: `</mfg/sd/description`

`one_line_string`

- Maximum length: 256 bytes
- One-line strings support a subset of `isascii` characters only. (Refer to the *ctype*(3) manpage.)
- No `isspace` characters, except for space and tab, are allowed.
- Examples: `Hewlett-Packard Company`

`path_mapping_string`

- Maximum length: none
- A value of the form: *`source`*`[=`*`destination`*`]` where the *`source`* defines the directory in which subsequently defined files are located. The optional *`destination`* maps the source to a destination directory in which the files will actually be installed.
- Examples: `/mfg/sd/files/usr = /usr`

`path_string`

- Maximum length: 255 bytes for tapes, 1024 bytes for depots
- An absolute or relative path to a file. Many attributes of this type are restricted to 255 bytes in length. This restriction is due to the *tar*(1) command, which requires a file's *basename(1)* be <= 100 bytes, and a file's *dirname(1)* to be <= 155 bytes. (Some implementations of tar enforce < and not <=.)
- Examples: `/usr` `/mfg/sd/scripts/configure`

`permission_string`

- Maximum length: none
- A value of the form:

  [`-m` *`mode`*| `-u` *`umask`*] [`-o` [*`owner`*[,]][*`uid`*]]
  [`-g` [*`group`*[,]][*`gid`*]]

  where each component defines a default permissions value for each file and directory defined in a fileset. The default values can be overridden in each file's specific definition. The owner and group fields are of type `tag_string`. The uid and gid fields are of type unsigned integer. The mode and umask are unsigned integers, but only supports the octal character set, 0-7.
- SD-UX will not override existing permissions based on this attribute if a file already exists on a target.
- Examples: `-u 0222 -o root -g sys`

`revision_string`

- Maximum length: 64 bytes
- Revision strings contain zero or more dot-separated `one_line_string` (above).
- Examples: `2.0, B.11.00`

`software_specification`

- Maximum length: none
- Software specifications are used to specify software in dependencies, ancestors and other attributes, as well as command line selections. This attribute uses the standard syntax for SD-UX `software_selections`. See "Software Selections" on page 56 for complete information.
- Examples: `SD.agent` or
  `SD,r=2.0,a=HP-UX_B.11.00_32`

`tag_string`

- Maximum length: 64 bytes
- Tag strings support a subset of `isascii()` characters only:

  - — Requires one or more characters from: “A-Z”, “a-z”, “0-9”, including the first character.
  - — The `isspace()` characters are not allowed.
  - — SDU metacharacters not allowed: `. , : =`
  - — Shell metacharacters not allowed: `# ; & () {} | < >`
  - — Shell quoting characters not allowed: `" ‘ ’ \`
  - — Directory path character (`/`) not allowed.
- Examples: `HP`, `SD-UX`

`uname_string`

- Maximum length: 64 bytes
- Uname strings containing a subset of *isascii()* characters only.
- No `isspace()` characters are allowed.
- Shell pattern matching notation allowed: [ ] * ? !
- Patterns can be “ORed” together using the separator: `|`
- Examples: `9000/7*:*|9000/8*:*`, `HP-UX, ?.11.*`

Product Specification File Semantics

The following sections describe how to specify a PSF and defines keywords.

Vendor-Defined Attributes You can create your own software attributes when packaging software.

Vendor-defined attributes are noted during packaging or when modified with swmodify. You can list these attributes with swlist.

When SD-UX encounters a keywords in a PSF that is not one of the standard keywords, the keyword and its associated values are preserved by being transferred to the INDEX or INFO files created by swpackage.

Nonstandard keywords are defined as a filename character string. The value associated with a keyword is processed as an `attribute_value`. It can be continued across multiple input lines or can reference a file containing the value for the keyword.

CAUTION If you misspell a standard keyword, SD-UX may mistake the keyword for a vendor-defined attribute, which may lead to packaging errors.

Distribution (Depot) Specification Distribution attributes let you list information about the media that will hold the depot (either tapes or CD/directory. (See "Depot Management Commands and Concepts" on page 134 for more information about depots.) Here is a PSF for a distribution:

```
distribution
  layout_version       1.0
  tag                  APPLICATIONS_CD
  copyright            < data/copyright.cd
  description          <data/description.cd
  number               B1234-56789
  title                HP-UX Applications Software Disk
# Optional vendor specification can be included.
# AT LEAST ONE PRODUCT SPECIFICATION MUST BE INCLUDED.
# Other product specifications are optional.
end
```

The `distribution` keyword is always required. All other attributes are optional.

`distribution` *or* `depot`	
	Keyword that begins the distribution specification. Each keyword defines an attribute of the distribution depot or tape itself. All keywords are optional, even if a distribution specification is included in a PSF.
`layout_version`	PSF syntax conforms to the `layout_version=1.0` of the POSIX 1387.2 Software Administration standard. Previous versions of SD-UX supported the POSIX `layout_version=0.8` syntax, which continues to be supported. (You can also select the layout version with the layout_version option for swpackage, swmodify, swcopy, or swlist.) See "Selecting the PSF Layout Version" on page 315 for more information.
`tag`	The short name of the target depot (tape) being created/modified by swpackage.
`copyright`	The text (or a pointer to a filename) for the copyright information for the depot's contents.
`description`	The description of the target depot; either the text itself or a pointer to a filename that contains the text.
`number`	The part or manufacturing number of the distribution media (CD or tape depot).
`title`	The full name of the target depot (tape) being created/modified by swpackage.
`end`	Ends the distribution specification, no value is required. This keyword is optional. If you use it and it is incorrectly placed, the specification will fail.

Vendor Specification The vendor attributes let you add a description to the PSF.

The `layout_version` defined for the PSF file determines how vendor specifications are associated with products and bundles. If a `layout_version` is not defined or is defined as 1.0, vendor specifications will be associated with all subsequent products and bundles that define a matching `vendor_tag` attribute.

If a `layout_version` of 0.8 is specified, all subsequent products and bundles will automatically be assigned to a `vendor_tag` from the last vendor object defined at the distribution level, if any, or from a vendor object defined within a product or bundle, unless a `vendor_tag` is explicitly defined.

The following is an example of a vendor specification:

```
vendor
 tag          HP
 description  < data/description.hp
 title        Hewlett-Packard Company
end
```

Each keyword defines an attribute of a vendor object. If a vendor specification is included in the PSF, swpackage requires the vendor and tag keywords.

NOTE The vendor specification is not the same as vendor-defined attributes. See “Vendor-Defined Attributes” on page 321 for more information.

`vendor`	Keyword that begins the vendor specification.
`tag`	Defines the identifier (short name) for the vendor.
`title`	Defines the full name (one line description) for the vendor.
`description`	Defines the multi-paragraph description of the vendor; the value is either the text itself (within double-quotes) or a pointer to the filename containing the text.
`end`	Ends the vendor specification. This keyword is optional.

Category Specification (Does not apply to layout version 0.8.) A software collection can contain a list of category objects that are used as a selection mechanism. Category objects are identified by the keyword "category" and contain additional information about the category. The `category_tag` attribute points to a particular category object and can appear anywhere within a product, bundle, subproduct, or fileset.

All software objects with the attribute of `is_patch` set to true are automatically assigned a category of "patch."

NOTE The `layout_version` keyword in the *`distribution class`* affects how categories are associated with products and bundles. See "Selecting the PSF Layout Version" on page 315 and "Product Specification File Semantics" on page 321 for more information.

The category specification looks like this:

```
category
   tag          patch_normal
   title        Category of patches
   description  For normal problems
   revision     0.0
end
```

Each keyword defines an attribute of the category object. If a category specification is included in the PSF, swpackage requires only the `category` and `tag` keywords.

`category`	Keyword that begins the category specification.
`tag`	The category short name identifier. Associates this object with a product or bundle. This `tag` attribute must match the `category_tag` attribute in the product or bundle.
`title`	A one-line string that defines the full name for the category.
`description`	A multi-line description of the category. The description value can consist of text or a filename for a text file.

`revision`	The revision information (release number, version). Determines which category object definition to maintain in a depot when a definition being installed or copied does not match a definition already in the depot with the same category tag.
`end`	An optional keyword that ends the specification. No value is required. If you place this keyword incorrectly in the PSF, the specification will fail.

Product or Bundle Specification The product specification is a required class in the PSF. It lets you identify the product you are packaging.

NOTE The `layout_version` keyword in the *`distribution class`* affects how category and vendor objects are associated with products and bundles. See “Selecting the PSF Layout Version” on page 315 and “Product Specification File Semantics” on page 321 for more information.

The product specification looks like this:

```
product
tag             SD
architecture    HP-UX_B.11.00_32/64
category_tag    systems_management
contents        prod.fsl,r=1.0,a=,v=
copyright       </mfg/sd/data/copyright
description     </mfg/sd/data/description
directory       /usr
is_locatable    false
is_patch        false
machine_type    *
number          J2326AA
os_name         HP-UX
os_release      ?.11.00.*
os_version      B.11.**
postkernel      /usr/lbin/kernel_build
+ readme        </mfg/sd/data/README
revision        2.0
title           Software Distributor
vendor_tag      HP

# Optional vendor specification
# Optional subproduct specification
# REQUIRED FILESET SPECIFICATION

end
```

For each product object specified, swpackage requires only the product and tag keywords, plus one or more fileset definitions. For each bundle specified, swpackage requires the `bundle`, `tag` and `contents` keywords.

`product`	Required keyword that begins the product specification.
`tag`	The product's identifier (short name).
`architecture`	The target system on which the product or bundle will run. Provides a human-readable summary of the four `uname` attributes (`machine_type`, `os_name`, `os_release` and `os_version`), which define the exact target system(s) the product supports.
`bundle`	Required keyword that begins the bundle specification.
`category_tag`	A repeatable tag-based attribute identifying a set of categories of which the software object is a member. This is used as a selection mechanism and can be used independent of patches. The default value is an empty list or `patch` if the `is_patch` attribute is set to `true`. Like `vendor_tag`, this attribute can be used as a pointer to a category object that contains additional information about the category (for example, a one-line title definition and a description of the category).

NOTE The category tag `patch` is reserved. When the `is_patch` product attribute is set to true, a built-in `category_tag` attribute of value `patch` is automatically included with the product definition.

`contents`	The list of `fully qualified` (all version distinguishing attributes included) software specs for the bundle.
`copyright`	A multi-line description of the product's copyright; either the text itself (in double quotes) or a pointer to the filename that contains the text.
`description`	A multi-paragraph description of the product; either the text itself (within double-quotes) or a pointer to the filename that contains the text.
`directory`	The default, absolute pathname to the directory in which the product's files will be installed (the root directory of the product). If not specified, swpackage assigns a value of /.

`is_locatable`	Defines whether a product or bundle can be installed to any product directory, or whether it must be installed into a specific directory. The attribute can be set to true or false. If not defined, swpackage sets the default attribute to “false.”
`is_patch`	A boolean flag that identifies a software object as a patch. The default value is false. When set to true, a built-in `category_tag` attribute of value `patch` is automatically included with the product definition.
`machine_type`	The system type on which the product will run. If not specified, the keyword is assigned a wildcard value of *, meaning it will run on all machines. If there are multiple platforms, you must separate each machine designation with a `|` (vertical bar). For example, a keyword value of **`9000/7*|9000/8*`** means the product will run on all HP Series 9000 Model 7XX or all HP 9000 Series 8XX machines. Alternatively, the value **`9000/[78]*`** would also work. Other examples: **`*`** **`(`**If not concerned with the machine type.) **`9000/7??:32*`** (Series 700, 32-bit capable hardware required) **`*:*64`** (64-bit capable hardware required_ **`*:32:`** (32-bit capable hardware required) **`9000/7??:*64`** (Series 700, 64-bit capable hardware required) **`9000/[78]??:32*`** (Series 800, 32-bit capable hardware required) **`9000/[78]??:*64`** (Series 800, 64-bit capable hardware required) The value is matched against a target’s **`uname -m`** or **`getconf _CS_HW_CPU_SUPP_BITS`** result.
`number`	The part or order number of the product.
`os_name`	The operating system name on which the product will run. If not specified, the attribute is assigned a value of *, meaning it will run on all operating systems. If there

	are multiple operating systems, use wildcards or the `|` symbol to separate them. The value is matched against a target's **`uname -s`** or **`getconf _CS_KERNEL_BITS`** result.
`os_release`	The release number of the product's operating system. If not specified, the attribute is assigned a value of `*`, meaning it will run on all operating systems. If there are multiple operating systems, use wildcards or the `|` symbol to separate them. The value is matched against a target's `uname -r` result.
`os_version`	The version number of the operating system(s) on which the product will run. If not specified, the attribute is assigned a value of `*`, meaning it runs on any version. If there are multiple operating systems, use wildcards or the `|` symbol to separate them. The value is matched against a target's `uname -v` result.
`postkernel`	Defines a kernel build script to be executed when kernel filesets are loaded. Kernel filesets have the `is_kernel` attribute set to `true`. The default kernel script is `/usr/sbin/mk_kernel`. (See the manual reference page for *mk_kernel* (1M) for more information.) The default script executes when the `postkernel` attribute is not specified. Only one kernel build script is allowed per product, and the script executes only once, even if defined for multiple filesets.
`readme`	A text file of the README information for the product. The value must be a pointer to the filename containing the text
`revision`	The revision information (release number, version) for the product or bundle.
`title`	A one-line string that further identifies the product or bundle.
`vendor_tag`	Associates this product or bundle with a vendor object defined separately in the PSF, if that object has a matching `tag` attribute.
`end`	Ends the product or bundle specification. No value is required. This keyword is optional. If you use it and it is incorrectly placed, the specification will fail.

Control Script Specification SD-UX supports execution of product and fileset control scripts that allow you to perform additional checks and operations with other HP-UX commands and functions. The swask, swinstall, swconfig, swverify, and swremove commands each can execute one or more control scripts on the primary roots. All scripts are optional but many times are needed correctly complete the task that you want your software package to perform. See Chapter 11, "Using Control Scripts," on page 369 for a complete discussion of control scripts.

Subproduct Specification The subproduct specification lets you group filesets within a larger product specification. Subproducts are optional. A subproduct specification looks like this:

```
subproduct
  tag          Manager
  contents     manager agent packager man doc
  description  </mfg/sd/data/manager/description
  title        SD Management Interfaces Subset
end
```

Each keyword defines an attribute of a subproduct object. If a subproduct object is specified, swpackage requires the `subproduct`, `tag`, and `contents` keywords.

`subproduct`	Keyword that begins a subproduct specification.
`tag`	The subproduct's identifier (short name).
`contents`	A whitespace-separated list of the subproduct's fileset `tag` values (that is, `contents` *`fileset1 fileset2 fileset3 ...filesetN`*). In the PSF, fileset definitions are not contained within subproduct definitions. The `contents` keyword is used to assign filesets to subproducts. This linkage allows a fileset to be contained in multiple subproducts.
`description`	A multi-line description of the subproduct; either the text itself (within double-quotes), or a pointer to the filename that contains the text.
`title`	A one-line string that further identifies the subproduct.
`end`	Ends the subproduct specification. No value is required. This keyword is optional. If you use it and it is incorrectly placed, the specification will fail.

Fileset Specification The fileset specification is required in the PSF. Use filesets to group files together.

A fileset specification looks like this:

```
fileset
   tag            manB
   ancestor       OLDSD.MAN
   architecture   HP-UX_B.11.00_32/64
   category_tag   manpg
   description    </mfg/sd/data/man/description
   is_kernel      false
   is_locatable   false
   is_patch       false
   is_reboot      false
   is_sparse      false
   machine_type   *
   os_name        HP-UX
   os_release     ?.11.00.*
   os_version     ?
   revision       2.40
   supersedes     product.fileset,fr=revision
   title          Commands (management utilities)
# Optional control script specification
# Optional dependency specification
# REQUIRED FILE SPECIFICATION
# Additional file specifications optional.
end
```

Each keyword defines an attribute as a fileset object. For each fileset object specified, swpackage requires the `fileset` and `tag` keywords, plus zero or more file specifications.

`tag`	The fileset identifier (short name).
`architecture`	Describes the target system(s) on which the fileset will run if filesets for multiple architecture are included in a single product. Provides a human-readable summary of the four `uname(1)` attributes which define the exact target system(s) the product supports. Many filesets do not include an architecture; only a product architecture need be defined.

`ancestor`	A list of filesets that will match the current fileset when installed on a target system, if the `match_target` installation option is specified. Also designates an ancestor fileset to check for when patch_match_target is defined.
`category_tag`	A repeatable tag-based attribute identifying a set of categories of which the software object is a member. This is used as a selection mechanism and can be used independent of patches. The default value is an empty list or patch if the `is_patch` attribute is set to true. Like `vendor_tag`, this attribute can be used as a pointer to a category object that contains additional information about the category (for example, a one-line title definition and a description of the category).

NOTE The category tag `patch` is reserved. When the `is_patch` file attribute is set to true, a built-in `category_tag` attribute of value `patch` is automatically included with the file definition.

`description`	Defines the multi-paragraph description of the fileset; the value is either the text itself (within double-quotes) or a pointer to the filename containing the text.
`is_kernel`	A value of true defines the fileset as being a contributor to the operating system kernel; the target system(s) kernel build process will be invoked after the fileset is installed. If this attribute is not specified, swpackage assumes a default value of false.
`is_locatable`	Defines whether a fileset can be installed to any product directory, or whether it must be installed into a specific directory. The attribute can be set to true or false. If not defined, swpackage sets the default attribute to false.
`is_patch`	Identifies a software object as a patch. The default value is false. When set to true, a built-in `category_tag` attribute of value patch is automatically included.

`is_reboot`	A value of true declares that the fileset requires a system reboot after installation. If this attribute is not specified, swpackage assumes a default value of false.
`is_sparse`	Indicates that a fileset contains only a subset of files in the base (ancestor) fileset and that the contents are to be merged with the base fileset. The default value is false. If the `is_patch` attribute is true, `is_sparse` is also set to true for the fileset, although it can be forced to false.
`machine_type`	The machine type on which the product will run. If not specified, the keyword is assigned a wildcard value of `*`, meaning it will run on all machines. If there are multiple machine platforms, you must separate each machine designation with a `|` (vertical bar). For example, a keyword value of **`9000/7*|9000/8*`** means the product will run on all HP Series 9000 Model 7XX or all HP 9000 Series 8XX machines. Alternatively, the value **`9000/[78]*`** would also work.

Other examples:

`*`	If not concerned with the machine type.
`9000/7??:32*`	Series 700, 32-bit capable hardware required.
`*:*64`	64-bit capable hardware required.
`*:32:`	32-bit capable hardware required.
`9000/7??:*64`	Series 700, 64-bit capable hardware required.
`9000/[78]??:32*`	Series 800, 32-bit capable hardware required.
`9000/[78]??:*64`	Series 800, 64-bit capable hardware required.

The value is matched against a target's **`uname -m`** or **`getconf _CS_HW_CPU_SUPP_BITS`** result.

`os_name`	Defines the operating system(s) on which the files will run if a fileset architecture has been defined. (If not specified, swpackage assigns a value of *, meaning the files run on all operating systems.) If there are multiple operating systems, use wildcards or use the '`|`' character to separate them. This attribute should pattern match to the value of **`uname -s`** or **`getconf KERNEL_BITS`** on the supported target systems.
`os_release`	Defines the operating system release(s) on which the files will run. (If not specified, swpackage assigns a value of *, meaning the files run on all releases.) If there are multiple operating system releases, use wildcards or use the '`|`' character to separate them. This attribute should pattern match to the value of `uname -r` on the supported target system(s).
`os_version`	The version number of the operating system(s) on which the product will run. If not specified, the attribute is assigned a value of `*`, meaning it runs on any version. If there are multiple operating systems, use wildcards or the `|` symbol to separate them. The value is matched against a target's `uname -v` result.
`revision`	Defines the revision (release number, version number) of the fileset.
`supersedes`	Used when a patch is replaced by (or merged into) a later patch. The attribute indicates which previous patches are replaced by the patch being installed or copied. This attribute value is a list of software specifications of other patches that this patch supersedes.
`title`	Defines the full name (one-line description) of the fileset.
`end`	Optional keyword to end the fileset specification. No value is required. If you place this keyword incorrectly, the file specification will fail.

Dependency Specification The swinstall, swcopy, swverify, and swremove commands recognize software dependencies. The default behavior for swinstall, for example, prevents an install unless all dependencies are met.

The PSF specifies dependencies between filesets. Dependencies are defined within the fileset class definition. (See “Fileset Specification” on page 331.)

You can also define dependencies between:

- A fileset and another product (namely, a subset of that product).
- A particular fileset within that product.
- The entire product.

SD-UX supports these types of dependencies:

`Corequisite`	Software that must be present for a fileset to operate correctly. For example, specifying a corequisite for an install fileset means that the corequisite must be installed or being installed when the fileset itself is installed. (Note that a corequisite dependency does not imply any “run-time dependency” (load order).)
`Exrerequisite`	Software that may not be present when the fileset is operated on by SD-UX. For example, specifying an exrequisite for a fileset prevents the fileset from being installed if any of the specified exrequisite software objects are installed or are being installed.
`Prerequisite`	Software that must be installed and/or configured correctly before a fileset can be operated on by SD-UX. Prerequisites control the order of an installation with swinstall (install-time dependency).

Dependencies are specified as a software_specification value type within the PSF. (See “PSF Value Types” on page 317 for more information.) For example:

```
corequisites   SD.data
prerequisites  productA,r>=2.1
exrequisites   productB,r>=2.1
```

NOTE A dependency must always be specified using a software specification that starts with the product tag for the requisite software.

You can specify multiple dependencies to define AND relationships between the dependencies (AND meaning that all dependencies must be satisfied).

You can also define OR relationships using the or (`|`) character. The following rules apply:

- White spaces are allowed around the OR character.
- OR dependencies are resolved from left to right.

Here is an example:

```
corequisite  P.F
prerequisite ProdA | ProdB | ProdC.F | ProdC.FS
corequisite  ProdX | ProdY | ProdZ | ProdW.FS
```

Control Script Specification SD-UX supports execution of product and fileset control scripts that allow you to perform additional checks and operations with other HP-UX commands and functions. The swask, swinstall, swconfig, swverify, and swremove commands each can execute one or more control scripts on the primary roots. You can write the scripts and include them in your software package. All scripts are optional but often are needed correctly complete the task that you want your software package to perform. See Chapter 11, “Using Control Scripts,” on page 369 for a complete discussion of control scripts.

File Specification Within a fileset specification, you can specify the following file types to be packaged into the fileset by swpackage:

- control script
- directory
- hard link
- regular file
- symbolic link
- archive

swpackage generates an error if the PSF contains an unrecognized or unpackageable file type.

The swpackage command supports specific mechanisms for specifying the files contained in a fileset:

default permission specification For all or some of the files in the fileset, you can define a default set of permissions.

directory mapping You can point swpackage at a source directory in which the fileset's files are located. In addition, you can map this source directory to the appropriate (destination) directory in which this subset of the product's files will be located.

explicit file specification For all or some of the files in the fileset, you can name each source file and destination location.

recursive (implicit) file specification If directory mapping is active, you can simply tell swpackage to recursively include all files in the directory into the fileset.

PSF extensions You can use **include** and **exclude** files to extend file definitions.

These mechanisms can all be used in combination with the others.

Default Permission Specifications By default, a destination file will inherit the mode, owner, and group of the source file. You can use the `file_permissions` keyword to set a default permission mask, owner, and group for all the files being packaged into the fileset:

`file_permissions` [`-m` *mode* | `-u` *umask*] [`-o` [*owner*[,]] [*uid*]]\
[`-g` [*group*[,]][*gid*]][`-t` *type*]

`file_permissions`

This keyword applies only to the fileset in which it is defined. You can specify multiple `file_permissions`; later definitions replace previous definitions.

`-m` *mode* — This option defines a default (octal) mode for all files.

`-u` *umask* — Instead of specifying an octal mode as the default, you can specify an octal *umask* (1) value that gets "subtracted" from an existing source file's mode to generate the mode of the destination file.

By specifying a umask, you can set a default mode for executable files, non-executable files, and directories. (A specific mode can be set for any file using `-m`.)

`-o` [*owner*[,]][*uid*]

This option defines the destination file's owner name and/or or uid. See the discussion of the `-o` option in "Explicit File Specification" on page 340 for more information.

`-g` [*group*[,]][*gid*]

This option defines the destination file's group name and/or or gid. See the discussion of the `-g` option in "Explicit File Specification" on page 340 for more information.

`-t` *type* — Defines files that need not exist before packaging.

The following examples illustrate the use of the `file_permission` keyword.

- Set a read only 444 mode for all file objects (requires override for every executable file and directory):

  ```
  file_permissions  -m 444
  ```

- Set a read mode for non-executable files, and a read/execute mode for executable files and directories:

  ```
  file_permissions  -u 222
  ```

- Set the same mode defaults, plus an owner and group:

  ```
  file_permissions  -u 222  -o bin  -g bin
  ```

- Set the same mode defaults, plus a uid and gid:

  ```
  file_permissions  -u 222  -o 2  -g 2
  ```

- Set the owner write permission in addition to the above:

  ```
  file_permissions  -u 022  -o 2  -g 2
  ```

- If you do not define `file_permissions`, swpackage uses the default value `file_permissions -u 000` for destination file objects based on existing source files. (Meaning the *`mode`*, *`owner/uid`*, *`group/gid`* are set based on the source file, unless specific overrides are specified for a destination file.)

Directory Mapping (Optional) The `directory` *`source`* `[=` *`destination`*`]` specification defines the source directory under which subsequently listed files are located. In addition, you can map the *`source`* directory to a *`destination`* directory under which the packaged files will be installed.

For example, the definition:

```
directory /build/hpux/mfg/usr = /usr
```

causes files from the `/build/hpux/mfg/` directory to have the prefix `/usr/sbin` when installed. The destination directory must be a superset of the product's `directory` attribute, if defined in the product specification. If the product's `directory` is defined, and the destination is not a superset, swpackage generates an error.

The destination directory must be an absolute pathname. If not, then swpackage generates an error.

The source directory can be either an absolute pathname, or a relative pathname. If relative, swpackage interprets it relative to the current working directory in which the command was invoked.

If the *`source`* directory does not exist, swpackage generates an error.

Explicit File Specification You can explicitly specify the files to be packaged into a fileset. If you want to recursively include all files and directories, use the recursive file specification (`file *`).

You can use the `directory` keyword to define a source (and destination) for explicitly specified files. If no `directory` keyword is active, then the full source path and the absolute destination path must be specified for each file. An explicit file specification overrides or adds to, on a file-by-file basis, the specifications set by the `directory` and/or `file_permissions` keywords.

An explicit file specification uses this form:

`file` [`-v`] [`-m` *`mode`*] [`-o` [*`owner`*[,]][*`uid`*]] [`-g` [*`group`*[,]][*`gid`*]]
[`-t` type] [*`source`*] [*`destination`*]

`file`	This keyword specifies an existing file (usually within the currently active source directory) to include in the fileset.
`source`	This value defines the path to a file you want to include in the package. If this is a relative path, swpackage will search for it relative to the source directory set by the `directory` keyword. If no source directory is active, swpackage will search for it relative to the current working directory in which the command was invoked. All attributes for the destination file object are taken from the source file, unless a `file_permission` keyword is active, or the `-m`, `-o`, or `-g` options are also included in the file specification.
`destination`	This value defines the destination path at which the file will be installed. If *`destination`* is a relative path, the active destination directory set by the directory keyword will be prefixed to it. If it is a relative path, and no destination directory is active, swpackage generates an error. If the destination is not specified, then the source path is used as the destination, with the appropriate mapping done with the active destination directory (if any).
`-m` *`mode`*	This option defines the (octal) mode for a file or directory at its destination.
`-o` *`[owner[,]][uid]`*	This option defines the file's owner name and/or uid at its destination. If only the *`owner`* is specified, then the owner and uid attributes are set for the destination file based on the packaging host's. If only the *`uid`* is specified, it is set as the destination's uid and no owner name is assigned. If both are specified, each sets the corresponding attribute for the file object.

During an installation, the owner attribute is used to set the owner name and uid, unless the owner name is not specified or is not defined in the target system's `/etc/passwd` file. In this case, the uid attribute is used to set the uid.

`-g` *`[group[,]][gid]`* This option defines the file's group name and/or gid at its destination. If only the *`group`* is specified, then the group and gid attributes are set for the destination file based on the packaging host's `/etc/group`. If only the gid specified, it is set as the destination's gid attribute and no group name is assigned. If both are specified, each sets the corresponding attribute for the file object.

During an installation, the group attribute is used to set the group name and gid, unless the group name is not specified or is not defined in the Target system's `/etc/group`. In this case, the gid attribute is used to set the gid.

`-t` *`type`* Defines a file of type `d` (directory), `s` (symbolic), `h` (hard link), or `a` (archive) for files that need not exist before packaging.

`-v` This option marks the file as volatile, meaning it can be modified (that is, deleted) after it is installed without impacting the fileset.

Files that may have their attributes (size, last modified time, etc.) changed through normal use after they are installed should be specified in the PSF file as volatile (by specifying `-v` on the line defining the file). swverify will not, by default, check file attributes for files that have the `is_volatile` attribute set to true (see the `check_volatile` option for swverify).

Error Messages

When processing existing files in a source directory, swpackage identifies the following four kinds of errors:

- Cannot search directory (permission denied)
- Cannot read the file (permission denied)
- Unsupported file type encountered (source file must be a control script, regular file, directory, hard link or symbolic link)
- File does not exist

Using Directory and File Keywords

The following examples illustrate the use of the `directory` and `file` keywords.

- Include all files under `/build/hpux/mfg` to be rooted under `/usr`:

  ```
  directory  /build/hpux/mfg=/usr
  file       *
  ```

- Include only certain files under `/build/hpux/mfg/`, to be rooted under `/usr` and `/var/adm/sw`:

  ```
  directory /build/hpux/mfg=/usr
  file      sbin/swinstall
  file      sbin/swcopy
  . . .
  directory /build/hpux/mfg=/var/adm/sw
  file      nls/swinstall.cat  nls/en_US.88591/swinstall.cat
  file      defaults newconfig/defaults
  file      defaults defaults
  ```

- Explicitly list files, no directory mapping specified:

  ```
  file  /build/hpux/mfg/usr/bin/swinstall /usr/sbin/swinstall
  file  /build/hpux/mfg/usr/bin/swcopy /usr/sbin/swcopy
  file  /build/hpux/mfg/data/nls/swinstall.cat
        /var/adm/sw/nls/en_US.88591/swinstall.cat
  file  /build/hpux/mfg/data/defaults
        /var/adm/sw/newconfig/defaults
  file  /build/hpux/mfg/data/defaults /var/adm/sw/defaults
  ```

- Use all specification types to include files:

  ```
  directory /build/hpux/mfg/usr=/usr
  file      *
  directory /build/hpux/mfg/data=/var/adm/sw
  file     defaults  newconfig/defaults
  file     /build/hpux/mfg/data/defaults=/var/adm/sw/defaults
  ```

Recursive File Specification The `file *` keyword directs swpackage to include every file (and directory) within the current source directory in the fileset. swpackage attempts to include the entire, recursive contents of the source directory in the fileset. (Partial wildcarding is not supported, e.g. `file dm*` to indicate all files starting with "dm".)

All attributes for the destination file object are taken from the source file, unless a `file_permission` keyword is active (this keyword is described below).

The user can specify multiple

```
directory  source[=destination]
file       *
```

pairs to gather all files from different source directories into a single fileset.

If you do not want to recursively include all files and directories, use the explicit file specification.

The `directory` keyword must have been previously specified before the `file *` specification can be used. If not, swpackage generates an error.

Error Messages

When processing the directory recursively, swpackage encounters the following errors:

- Cannot search directory (permission denied)
- Cannot read the file (permission denied)
- Unsupported file type encountered

PSF Extensions A PSF can contain extended file definitions. SD currently supports exclude and include files.

Exclude files let you explicitly exclude files that would otherwise be included in the PSF. The syntax is:

exclude filename

An exclude file can only be specified after a file definition. The file listed after the exclude keyword is excluded from the current context (for example, from a recursive file definition or wildcard).

If the filename specifies a directory, then all files below that directory are excluded.

Include files let you include file definitions from a separate file. The syntax is:

file < filename

The include file must be separated from the file keyword by a less than sign (<).

Re-Specifying Files

In addition to being able to specify files as a group (with `file *`) for general attributes, the PSF also allows you to “re-specify” files *within* that definition to modify individual attributes.

For example, suppose you wanted to specify all the files in a fileset which contained 100 files. All these files were to be recursively “discovered” and packaged into the fileset. Most of them would have the same owner, group, and mode (and other file attributes).

Out of those 100 files, there might be five that are volatile (that is, you don’t care if they get modified or deleted). So, instead of listing all 100 files individually, and using the `-v` option for the five, you could specify all 100 with `file *` and then modify the five individually in their own way. For example, with files `1`, `2`, `3`, `4`, and `5`:

```
directory source = /product file *

   file -v 1
   file -v 2
   file -v 3
   file -v 4
   file -v 5
```

This also works well for permissions. For example, assume that nearly all the 100 files in the preceding example had the same permission attributes, but files `1`, `2`, and `3` required a different owner and mode:

```
directory  source = /product

   file_permissions -o bin -g bin -m 555
   file *

   file_permissions -o root -g other -m -04555
   file 1
   file 2
   file 3
```

This capability combines the recursive file specifications function with explicit file specification. (See “Explicit File Specification” on page 340).

Packaging the Software (swpackage)

The swpackage command packages software products defined in a PSF into a depot. You can then use the software in the depot with other SD-UX commands.

Overview

Features and limitations include:

- Uses the PSF to organize files into products, subproducts, and filesets.
- Can include control scripts and PSFs to further specify how to handle the software when installing it onto the target system.
- Sets permissions of the files being packaged.
- Can package either simple, one-fileset products or complex products with many filesets and subproducts.
- Provides a way to repackage (change) existing products.
- The swpackage command provides only a command line user interface. There is no Graphical User Interface for the packaging tasks.
- Can create directory depots (including CDs) or tape depots (useful for distributing software via the internet).
- Does not automatically register newly created depots. You must use the swreg command (see “Registering and Unregistering Depots (swreg)” on page 151).

The swpackage Process

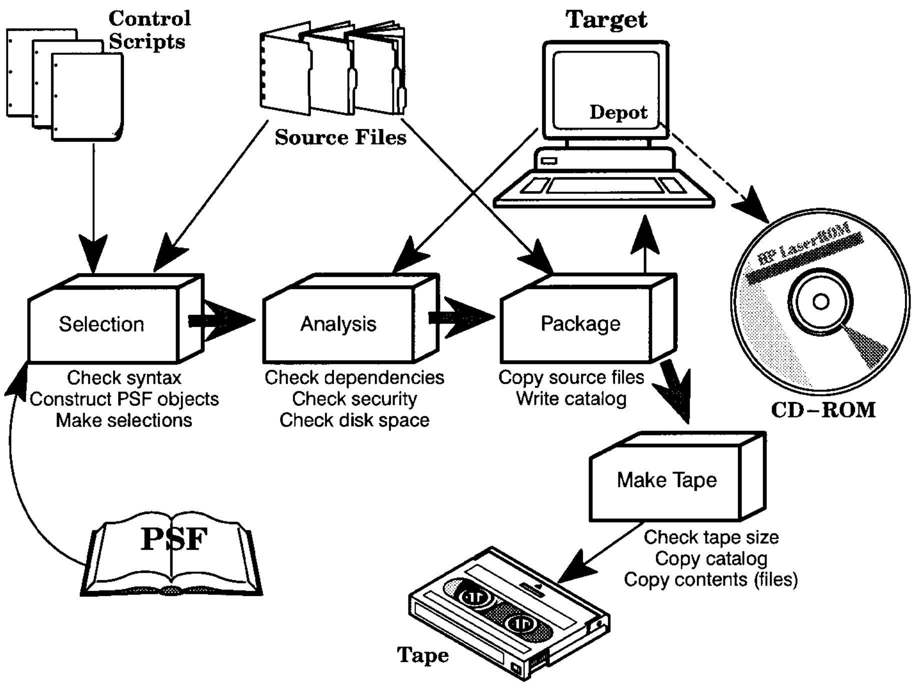
The swpackage process includes up to four phases:

Table 10-4 swpackage Process Phases

I. Selection	swpackage reads the PSF
II. Analysis	swpackage analyzes the packaging tasks and requirements before actually packaging the software to the target depot or tape. swpackage compares the software to be packaged against the target depot to make sure the packaging operation will be successful.
III. Build	swpackage packages the source files and information into a product object, and inserts the product into the distribution depot. swpackage creates the depot but does not register it. You must have appropriate SD-UX permission to create this new depot on the local host. If the target (destination) is a tape media, a temporary depot is created.
IV. Make Tape	(Optional) This phase occurs only if you are packaging to a distribution tape. swpackage copies the source files and a temporary depot catalog to the tape. (Note that swpackage cannot compress files when writing to a tape.)

Figure 10-1, “An Overview of the Packaging Process,” shows an overview of the swpackage session.

Figure 10-1 An Overview of the Packaging Process



Phase I: Selection When you run swpackage, you must specify a PSF and any other options you wish to include. The swpackage command begins the session by telling you the source, target, software selections, and options used,

- Determine the product, subproduct, and fileset required for the structure
- Determine which files are contained in each fileset
- Determine the attributes associated with each objects
- Check PSF syntax and terminates the session if any are encountered

Phase II: Analysis swpackage performs four checks during this phase:

1. **Check for unresolved dependencies.**

   For every fileset in each selected product, swpackage checks to see if a requisite of the fileset is *not* also selected or *not* already present in the target depot. Unresolved dependencies within the product generate errors. Unresolved dependencies across products produce notes.

2. **Check your authorization to package (or re-package) products.**

   For each new product (a product that does *not* exist on the target depot) swpackage checks the target depot to see if you have permission to create a new product on it (insert permission). If you do not, the product is not selected.

   For each existing product (one you are re-packaging) swpackage checks to see if you have permission to change it (write permission). If you do not, the product is unselected.

   If all products are not selected because permission is denied, the session terminates with an error.

   If the depot is a new depot or if you are packaging to a tape, this authorization check is skipped. If you have permission to create a new depot, then you have permission to create products within it. Since a tape session first writes to a temporary depot then copies it to tape, if you have permission to create a new (temporary) depot, you can package to tape.

3. **Check for software being repackaged.**

   For each selected product, swpackage checks to see if the product already exists in the target depot.

   - If it does exist, swpackage checks to see which filesets are being added (new filesets) or modified.
   - If it exists and all filesets are selected, swpackage checks to see if any existing filesets have been obsoleted by the new product.

4. **Performing Disk Space Analysis (DSA)**

   swpackage verifies that the target depot has enough free disk space to package the selected products.

   - If adequate disk space is available for the packaging operation to proceed, swpackage writes a note to the log file to note the impact on disk space.
   - An error results if the package will encroach into the disk's minfree space.
   - An error results if the package phase requires more disk space than is available.
   - If you set the `enforce_dsa` command option to false, swpackage changes disk space errors to warnings and continues. This lets you cross into the minfree space to complete a packaging operation.

Phase III: Build

When packaging a product, if the target depot does not exist, swpackage creates it. If it does exist, swpackage will merge new product(s) into it. For each different version of the product, a directory is created using the defined product tag attribute and a unique instance number (instance ID) for all the product versions that have the same tag.

Before a new storage directory is created, swpackage checks to see if this product version has the same identifying attributes as an existing product version.

If all the identifying attributes match, you are re-packaging (modifying) an existing version. Otherwise, swpackage creates a new version in the target distribution.

The packaging process uses an explicit ordering to avoid corrupting the target distribution if a fatal error occurs. Each product is packaged in its entirety and when all specified products have been packaged successfully, the distribution's global INDEX file is built/rebuilt. Within each product construction, the following order is adhered to:

1. Check if the product is new or already exists. If it is new, create the product's storage directory.
2. For each fileset in the product, copy the fileset's files into their storage location (within the product's storage directory), and create the fileset's catalog (database information) files.

3. After the individual filesets, create the product’s informational files (meta-files).

A target depot is only the first step in creating a CD-ROM. If the ISO 9660 standard format is desired, a utility to perform this conversion would be necessary. This conversion is not supported by swpackage.

Distribution tapes are created in tar format (although SD-UX commands can also read depots from cpio format tapes). To create the tape, swpackage first builds the products into a temporary distribution depot. (The depot is removed when swpackage completes.) To conserve space, all files exist as references to the real source files. After the distribution depot is constructed, swpackage then archives it, along with the real files, onto the tape device.

When archiving a product that contains kernel filesets onto a tape media, swpackage puts these filesets first within the archive to provide efficient access by swinstall. swpackage also orders filesets based on prerequisite dependency relationships.

Phase IV: Make Tape

This optional phase occurs only when you package to a distribution tape.

- In this phase, swpackage copies the source files and a temporary depot catalog to the tape.
- swpackage does a tape space calculation to ensure that the tape can hold the software package. If one tape cannot hold it all, then swpackage will partition the software across multiple tapes.
- swpackage cannot compress files when writing to a tape.

Using swpackage

swpackage Syntax

swpackage [`-p`] [`-v`] [`-V`] [`-C` *`session_file`*]
[`-d` *`directory|device`*] [`-f` *`software_file`*]
[`-s` *`product_specification_file|directory`*]
[`-S` *`session_file`*] [`-x` *`option=value`*] [`-X` *`option_file`*]
[*`software_selections`*] [`@` *`target_selection`*]

Options and Operands

`-p`
Previews the specified package session without actually creating or modifying the depot or tape.

`-v`
Turns on verbose output to `stdout` and lists messages for each product, subproduct and fileset being packaged. (The swpackage logfile in `/var/adm/sw/swpackage.log` is not affected by this option.)

`-V`
List the data model revisions which swpackage can read. swpackage always packages using the latest data model revision.

`-C` *`session_file`*
Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.

`-d` *`directory|device`*
If creating a distribution directory, this option defines the pathname of the *`directory`*.

If creating a distribution tape, this option defines the *`device`* file on which to write the distribution. When creating a distribution tape, the tape device (file) must exist, and the `target_type=tape` option must be specified.

`-f` *`software_file`*
Read a list of software selections from a separate file instead of (or in addition to) the command line. See “Software Files” on page 58.

`-s` *`product_specification_file|directory`*
Specifies the PSF to use or the existing directory to use as the source for the packaging session.

`-S` *`session_file`*
: Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-x` *`option=value`*
: Sets a command *`option`* to *`value`* and overrides default values or a values in options files. See “Changing Command Options” on page 353.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 353.

`software_selections`
: The software objects to be installed. See “Software Selections” on page 56.

If you do *not* include this specification, swpackage packages all the products listed in the PSF.

`@` *`target_ selections`*
: The target of the command. See “Target Selections” on page 58.

If you are creating a distribution depot (directory), this operand defines the location of the *`directory`*. Without this operand, `/var/spool/sw` is used as the default depot directory.

If you are creating a distribution tape, this operand names the *`device`* file on which to write the `tar` archive. swpackage must be able to determine if the media is a DDS tape or a disk file. Without this operand, swpackage uses the device file, `/dev/swtape`.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the options and default values that apply to swconfig.

Table 10-5 swpackage Command Options and Default Values

- admin_directory=/var/spool/sw
- allow_partial_bundles=true
- compress_command= /usr/contrib/bin/gzip
- compress_files=false
- compress_index=false
- compression_type=gzip
- create_target_acls=true
- distribution_source_directory= /var/spool/sw
- distribution_target_directory= /var/spool/sw
- distribution_target_serial= /dev/rmt/0m
- enforce_dsa=true
- follow_symlinks=false
- include_file_revisions=false
- layout_version=1.0
- log_msgid=0
- logdetail=false
- logfile=/var/adm/sw/swpackage.l og
- loglevel=1
- media_capacity=1330
- media_type=directory
- package_in_place=false
- reinstall_files=true
- reinstall_files_use_cksum=true
- run_as_superuser=true
- software=
- source_files=psf
- source_type=directory
- targets=
- uncompress_cmd=
- verbose=
- write_remote_files=false

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

Output of Logfile Messages

The log file /var/adm/sw/swpackage.log captures output from the swpackage session.

- Message logging by default sends verbose messages to stdout.

  (Setting the verbose option to 0 reduces the amount of information in stdout.)
- Message logging also sends errors and warnings to stderr.
- No logfile messages are written in preview (-p) mode.
- The logfile is equal to stdout plus stderr.

Here is a sample log:

```
=======  01/27/01 18:58:45 MST  BEGIN swpackage SESSION
       * Session started for user "root@sdtest.myco.com".

       * Source:        vewd:test.psf
       * Target:        vewd:/var/spool/sw
       * Software selections:
             *
       * Options:
            preview            true
            verbose            1
            loglevel           1
            logfile            /var/adm/sw/swpackage.log

            source_type              file
            target_type              directory

            package_in_place         false
            follow_symlinks          false
            include_file_revisions      false
            enforce_dsa                 true
            reinstall_files             true
            reinstall_files_use_cksum   false
            write_remote_files          false
            create_target_acls          true
```

```
    * Beginning Selection Phase.
    * Reading the Product Specification File (PSF) "test.psf
".
    * Reading the product "SD" at line 1.
    * Reading the fileset "commands" at line 4.
=======  01/27/01 18:58:45 MST  END swpackage SESSION
```

Packaging Tasks and Examples

To package the software products defined in the PSF `product.psf` into the distribution depot `/var/spool/sw` and preview the task at the verbose level before actually performing it, type:

```
swpackage -p -v -s product.psf @ /var/spool/sw
```

Registering Depots Created by swpackage

When a new depot is created by swpackage, it is *not* automatically registered with the local host's swagentd daemon.

To verify that the depot is registered, type:

```
swlist -l depot @ MyDepot
```

To register the depot, you must execute the swreg command:

```
swreg -l depot depot_to_register
```

Registering a depot makes it generally available as a source for swinstall and swcopy tasks.

Registration provides a type of public recognition for the packaged depot:

- You can see the depot in the swinstall/swcopy GUI and see it in swlist depot-level listings.
- You can read products from the depot (for example, to install).

For more information about registering depots, see "Registering and Unregistering Depots (swreg)" on page 151.

NOTE If the only use of a depot created with swpackage is local access by the packaging user, depot registration is not required.

Creating and Mastering a CD-ROM Depot

When swpackage creates a new depot or packages a new product, it always creates an ACL for the depot/product. If you were to create a depot and then master it onto a CD-ROM, the CD-ROM would contain all those ACLs, which could cause the following problems:

- it may result in too-restrictive permissions on the CD-ROM depot.
- you could have too many user-specific ACLs on the CD-ROM.

To solve these problems, you can tell swpackage to *not* create ACLs in the depot by setting the `create_target_acls` option to false.

This feature is provided only for the superuser because only the local superuser can change, delete, or add ACLs to a depot that has no ACLs. The local superuser always has all permissions.

Setting the `create_target_acls` to false causes swpackage to skip the creation of ACLs for each new product being packaged (and for the depot, if it is new). This option has no impact on the ACLs that already exist in the depot.

When a depot is used as a source for other SD-UX operations, its ACLs (or lack of ACLs) have no bearing on the ACLs created for the targets of the operation. Source ACLs are not related to target ACLs.

The swpackage command never creates ACLs when software is packaged onto a tape.

Compressing Files to Increase Performance

The packaging process may pass large amounts of data back and forth over the network and might slow down network performance. The `compress_files` option can improve performance by first compressing files that are to be transferred. This performance gained depends on the type of files transferred. Binary files compress less than 50%, text files generally compress more. Improvements are best when transfers are across a slow network (approximately 50Kbytes/second or less).

If set to true, `compress_files` compresses files (if they have not been compressed previously by SD-UX) before transfer from a source. You may also specify a compression type with the `compression_type` option or specify a compression command with the `compression_command` option.

This option should be set to true only when network bandwidth is clearly restricting total throughput. If it is not clear that this option will help, compare packaging operations both with and without compression before consistently using this option. See Appendix A, "Command Options," on page 421 for more information on using command options.

NOTE swpackage cannot compress files when writing to a tape.

Packaging Security

SD-UX provides Access Control Lists (ACLs) to authorize who has permission to perform specific operations on depots. Because the swpackage command creates and modifies local depots only, the SD-UX security provisions for remote operations do not apply to swpackage. See Chapter 9, "SD-UX Security," on page 255 for more information on ACLs.

The swpackage command operates as `setuid root`, that is, the Package Selection phase operates as the invoking user, the Analysis and Packaging phases operate as the superuser. The superuser owns and manages all depots and therefore has all permissions for all operations on a depot. If the depot happens to be on an NFS volume, access problems will not arise from ACLs, but will arise if the local superuser does not have NFS root access on the NFS mounted file system.

If you are not the local superuser, you will not have permission to create or modify a depot unless the local superuser grants you permission.

swpackage checks and enforces the following permissions:

1. **Can you create a new depot?**

Superuser	Yes
Other	Yes, if the ACL for the local host grants the user "insert" permission, i.e. permission to insert a new depot into the host. If the proper permissions are not in place and the depot is a new one, swpackage terminates with an error.

2. **Can you create a new product?**

Superuser	Yes
Other	Yes, if the depot is new and you passed check #1 above or if the ACL for an existing depot grants you insert permission, i.e. permission to change the contents of the depot (by adding a new product). If you are denied authorization to create a new product, swpackage generates an error message and excludes the product from the session.

3. **Can you modify an existing product?**

Superuser	Yes
Other	Yes, if the ACL for the existing product grants you write permission, i.e. permission to overwrite/change the contents of the product. If you are denied authorization to change an existing product, swpackage generates an error message and excludes the product from the session. If you are denied insert and write permission for all selected products, swpackage terminates with an error.

4. **Can you change the depot-level attributes?**

Superuser	Yes

Other	Yes, if the depot is a new one and you passed check #1 above or if the ACL for an existing depot grants you write permission, i.e. permission to write/change the contents of the depot (same as #2 above). If you are denied authorization to change an existing depot, and if the PSF specifies some depot-level attributes, then swpackage produces a warning message and does not change the depot attributes.

ACL Creation

When swpackage creates a new depot or a new product, it also creates an ACL for it:

New depot swpackage creates an ACL for the depot and a template ACL for all the products that will be packaged into it.

The depot ACL is generated from the host's `global_soc_template` ACL (that is, the template ACL established for new depots and new root file systems).

The depot's `product_template` ACL is generated from the host's `global_product_template` ACL (that is, the host's template ACL for new products).

The user running swpackage is established as the owner of the new depot and is granted permissions as defined in the depot ACL (which come from the `global_soc_template`).

New product swpackage creates an ACL for the product; the ACL is generated from the depot's `product_template` ACL.

ACL creation can be disabled by setting the create_target_acls command to false.

When no ACL exists for a depot, only the superuser can create new products or add/modify depot attributes. When no ACL exists for a product, only the superuser can modify it.

Repackaging or Modifying a Software Package

There are two types of repackaging:

1. Adding to or modifying a fileset in an existing product.

   - Editing the PSF by adding a new fileset definition or changing an existing fileset's definition.
   - Running swpackage on the edited PSF, specifying the new/changed fileset on the command line:

```
swpackage -s psf <other options> \
          product.fileset @  depot
```

   This invocation works regardless of whether subproducts are defined in the product.

   - If you change a fileset by changing its tag attribute, swpackage cannot correlate the existing, obsolete fileset with the new fileset. Both become part of the changed product. To get rid of the obsolete (renamed) fileset, use swremove:

```
swremove -d  product.old_fileset @ depot
```

2. Modifying an entire existing product.

   - Editing the PSF by adding new fileset definitions, changing existing fileset definitions, deleting existing fileset definitions or changing the product's definition (product-level attributes).
   - Running swpackage on the PSF, specifying the product on the command line:

```
swpackage -s psf <other options> product @ depot
```

   - If you have deleted some fileset definitions in the PSF or modified a fileset by changing it's tag attribute, swpackage will produce warning messages about the existing filesets that are not part of the modified product's definition (in the PSF). The existing filesets plus the new filesets in the product's definition (in the PSF) will all be contained in the modified product.

     The warnings are produced during analysis phase, and are only produced when the whole product is being repackaged (as opposed to subsets of the product).

- To get rid of the obsolete (renamed) filesets, use swremove:

```
swremove -d product.old_fileset @ depot
```

- You may want to swremove the product entirely before repackaging the changes:

```
swremove -d product @ depot
swpackage -s psf <other options> product @ depot
```

Packaging In Place

If you set the `package_in_place` option to true, swpackage packages each of the specified products such that the source files are not copied into the target depot. Instead, swpackage inserts references to the source files that make up the contents of each fileset. Control scripts are always copied.

This feature lets you package products in a development or test environment without consuming the full disk space of copying all the source files into the target depot. Disk space analysis is skipped when the `package_in_place` option is true.

The source files must remain in existence. If some are deleted, any operations that use the depot as a source (for example, installing the product with swinstall) will fail when they try to access the missing source files.

If a source file changes and the product is not repackaged, the information that describes the source file will be incorrect (for example, the file `checksum`). This incorrect information will not prevent the use of that target depot as a source (for example, installing with swinstall). However, the incorrect information will be propagated along each time the product is copied or installed from the depot. The result is that a swverify operation on the installed product always flags the inconsistencies with an error unless you disable the check of file contents.

Following Symbolic Links in the Source

If you set the `follow_symlinks` option to true, swpackage follows every source file that is a symbolic link and include the file it points to in the packaged fileset.

swpackage also follows each source directory that is a symbolic link, which affects the behavior of the `file *` keyword (recursive file specification). Instead of including just the symbolic link in the packaged fileset, the directory it points to and all files contained below it will be included in the packaged fileset.

The default value for this option is false, which causes symbolic links that are encountered in the source to be packaged as symbolic links. The symbolic link can point to a file that is also part of the fileset, or to a file that is not.

Generating File Revisions

If you set the `include_file_revisions` option to true, swpackage examines each source file using the what and ident commands to extract an SCCS or RCS revision value and assign it as the file's revision attribute.

Because a file can have multiple revision strings embedded within it, swpackage uses the first one returned. It extracts the revision value from the full revision string and stores it.

This option is time consuming, especially when a `what` search fails and the ident command is then executed.

The default value for this option is false, which causes swpackage to skip the examination. No value for the revision attribute is assigned to the files being packaged.

Depots on Remote File Systems

Because the swpackage analysis and build phases operate as the superuser, there are constraints on how swpackage creates, adds to, or modifies products on a depot that exists in an NFS-mounted file system.

If the superuser does not have write permission on the remote file system, swpackage will be unable to create a new depot-it will terminate before the analysis phase begins.

If the superuser *does* have write permission on the remote file system but the option `write_remote_files` is false, swpackage will be unable to create a new depot - it will terminate before the analysis phase begins.

If the superuser *does* have write permission on the remote file system and you set the `write_remote_files` to true, swpackage creates the new depot and package products into it.

The constraints for an existing NFS mounted depot are the same as when creating a new depot.

So, you must:

1. Set the `write_remote_files` option to true and
2. Make sure the superuser can write to the NFS file system to package a depot on an NFS-mounted file system.

When these constraints are satisfied, the ACL protection mechanism controls operations on NFS mounted depots the same way it controls operations on local depots.

Verifying the Software Package

If swpackage created a depot rather than storing the package in an existing registered depot, you must register the depot with the swreg command. (See “Registering Depots Created by swpackage” on page 356.)

After the depot is registered, you can verify it with the swverify command. For example, to verify the integrity of the product Pascal in the local default depot:

```
swverify -d Pascal
```

For more information about verifying depots, see “Verifying a Depot (swverify -d)” on page 161.

You can also test the package by installing it on a system. For example, to install the package named `Pascal`, located on the default depot `/var/spool/sw` in the host `svrhost`, onto the primary root of a host named `myhost`:

```
swinstall -s svrhost Pascal @ myhost
```

(This example does not specify the depot location because it is assumed that the software is located in the default `/var/spool/sw` on `svrhost`.)

For more information about verifying installed software, see “Verifying Your Installation (swverify)” on page 89

Packaging Patch Software

A number of software attributes are available to all software levels (bundles, products, subproducts, and filesets) that permit packaging of patch software. For complete information on patch attributes and a sample PSF, see Chapter 5, “Managing Patches,” on page 163.

Writing to Multiple Tapes

When you package products to a distribution tape, the `media_capacity` option defines the size of the tape media (in one million byte units). The default value for this option is `media_capacity=1330`, which is the size of an HP DDS tape. If the target tape is not a DDS tape, you must specify the `media_capacity` value.

NOTE The capacity of the DDS tape is in one million byte units (1,000,000 bytes), *not* Mbyte units (1,048,576 bytes). Most tape drive manufacturers specify capacity in one-million byte units.

If the products being packaged require more space than the specified media capacity, swpackage will partition the products across multiple tapes.

To find out if multiple tapes will be required, swpackage will calculate the tape blocks required to store the depot catalog and each product's contents.

When multiple tapes are necessary, swpackage writes the entire catalog onto the first tape plus any product contents that also fit. For each subsequent tape, swpackage prompts you for a “tape is ready” response before continuing.

To continue with the next tape, enter one of the following responses:

Return	Use the same device.
pathname	Use the new device/file *`pathname`*.
quit	Terminate the write-to-tape operation.

Partitioning is done at the fileset level, so a product can span multiple tapes. A single fileset's contents cannot span multiple tapes. If any single fileset has a size that exceeds the media capacity, swpackage generates an error and terminates. It also generates an error if the catalog will not fit on the first tape.

Making Tapes from an Existing Depot

You can copy one or more products from an existing depot to a tape using swpackage. Instead of specifying a PSF as the source for a packaging session, just specify an existing depot. For example:

```
swpackage -s /var/spool/sw  ...
```

To copy all of the products in a depot to a tape:

```
swpackage -s depot -d tape -x target_type=tape
```

To copy only some of the products in a depot to a tape, specify the products as software selections:

```
swpackage -s depot -d tape -x target_type=tape \
   product1 product2 ...
```

You can also use the `-f` *`file`* option can be used to specify several software selections instead of listing them on the command line.

When products are copied from a depot to a tape, the ACLs within the depot are *not* copied. (The swpackage command never creates ACLs when software is packaged onto a tape.)

swpackage cannot compress files when writing to a tape.

11 Using Control Scripts

This chapter discusses how to use control scripts.

Table 11-1 Chapter Topics

Introduction to Control Scripts

SD-UX supports execution of both product and fileset **control scripts**. These shell scripts allow you to perform additional, customized checks and operations as part your regular software management tasks. The swinstall, swconfig, swverify, swask, and swremove commands can execute one or more of these scripts. Control scripts are usually supplied by software vendors, but you can write your own. All control scripts are optional.

Product level control scripts are run when any fileset within that product is selected for installation, configuration, verification, or removal so the activities in product control scripts must pertain to all filesets in that product, but not to any fileset in particular. Actions you want to apply to every fileset in a product should be in the appropriate product level control script.

Fileset scripts must pertain only to the installation, configuration, or removal of that fileset, and not to any other fileset or to a parent product.

Control scripts can perform a wide variety of customization and configuration tasks, such as (but not limited to):

- Verifying if someone is actively using the product and, if so, preventing reinstallation, update or removal.
- Ensuring the local host system is compatible with the software (scripts can check beyond the compatibility enforced by the product's `uname` attributes).
- Removing obsolete files or previously installed versions of the product.
- Creating links to, or additional copies of, files after they have been installed.
- Copying configurable files into place on first-time installation.
- Conditionally copying configurable files into place on later updates.
- Modifying existing configuration files for new features.
- Rebuilding custom versions of configuration files.
- Creating device files or custom programs.
- Killing and/or starting daemons.

Types of Control Scripts

Here are the control scripts that SD-UX supports:

- **Checkinstall Script**

  This script is run by swinstall during its Analysis phase to insure that the installation (and configuration) can be attempted. For example, the OS run state, running processes, or other prerequisite conditions beyond dependencies could be checked. It should not change the state of the system.

  A checkinstall script's chief merit is its ability to detect if the system contains a hardware configuration that might lead to catastrophe - an unbootable system or file system corruption - if the installation of the selected software was allowed to proceed. It also acts as the test for conflicts with other software selections or with software already installed.

- **Preinstall Script**

  This script is run by swinstall before loading the software files. For example, this script could remove obsolete files, or move an existing file aside during an update.

  A preinstall script is called during swinstall's Execution Phase. The preinstall script for each file is executed just before that fileset's files are installed onto the target system. A product level preinstall script is called before a product's filesets.

  Preinstall scripts for all kernel filesets and their prerequisites are all run before the kernel build takes place. If the kernel build fails and swinstall exits, the preinstall scripts are removed from the system. Product level preinstall scripts are invoked twice for all products that contain kernel filesets: once when the kernel filesets are their prerequisites are installed; a second time when the remaining filesets are installed.

- **Postinstall Script**

  This script is run by swinstall after loading the software files. For example, this script could move a default file into place.

  The postinstall script is part of swinstall's Load phase. After the files are loaded, the fileset's postinstall script is run. Then, the products's postinstall script (if any) is run.

- **Unpreinstall Script**

  Unpreinstall scripts are executed during the load phase of swinstall if recovery is initiated.

  All undo scripts are executed in the reverse order of the normal scripts. For each fileset being recovered, the unpostinstall script is run, the fileset files are restored, and the unpreinstall script is run. An undo script is executed if its corresponding script was executed.

  An unpreinstall script should undo any operation that the preinstall script did. For example, if the preinstall script moved a file, the unpreinstall script should move it back. If the preinstall script copied a file, the unpreinstall script should remove it.

  For a product to be recoverable, no files should be removed by preinstall or postinstall scripts. Configure scripts are a good place to remove obsolete files.

  A product unpreinstall script is run after the fileset unpreinstall scripts.

- **Unpostinstall Script**

  Unpostinstall scripts are executed during the load phase of swinstall if recovery is initiated.

  All undo scripts are executed in the reverse order of the normal scripts. An undo script is executed if its corresponding script was executed.

  An unpostinstall script should undo any operation that the postinstall script did. For example, if the postinstall script moved a file, the unpostinstall script should move it back. If the postinstall script copied a file, the unpostinstall script should remove it.

  For a product to be recoverable, no files should be removed by preinstall or postinstall scripts. Configure scripts are a good place to remove obsolete files.

NOTE Product level unpostinstall scripts are not supported.

- **Configure Script**

  This script is run by swinstall or by swconfig to configure the host for the software, or configure the software for host-specific information. For example, this script could change a host's specific configuration file such as `/etc/services`, add the host name or other host resources such as available printers to its own configuration file, or perform compilations.

  Configure scripts are run by swinstall for all products (in prerequisite order) *after* the products have completed the Load phase. However, they are only run when installing to a system that will actually be using the software. They are deferred when installing to an alternate root (for example, for diskless or building test file systems) and run instead by the swconfig command when the alternate root is now the root of the system using the software.

  The swconfig command can also be used to rerun configure scripts that failed during a normal install. A successful execution of the configure step (whether there is a script or not) moves the software from the installed state to the configured or ready-to-use state. Configure scripts (and all others) must be able to be run many times (that is, they must be re-executable).

  Configure scripts are a good place to remove obsolete files.

  Configure scripts are not run for installations to alternate roots.

- **Verify Script**

  Verify scripts are run by the swverify command any time after the software has been installed and configured. Like other scripts, they are intended to verify anything that the SD-UX software management tools do not verify by default. For example, this script could check to see that the software is configured properly and that you have a proper license to use it.

- **Fix Script**

  Defines the fix script run by swverify to correct and report problems on installed software. The fix script can create missing directories, correct file modifications (mode, owner, group, major, and minor), and recreate symbolic links.

- **Unconfigure Script**

  A script run by swconfig or swremove to undo a host or software configuration originally performed by a configure script. For example, an unconfigure script could remove the configuration from the `/etc/services` file. (The unconfigure task moves the software from the configured state back to the installed state.)

  Only the swremove command actually removes software. although you can run unconfigure scripts using swconfig. Unconfigure scripts are not run for removals from alternate roots.

- **Checkremove Scripts**

  The checkremove script is run by swremove during the remove analysis phase to allow any checks before the software is permanently removed. For example, the script could check whether anyone was currently using the software before removing it.

- **Preremove Scripts**

  This script is executed just before removing files. It can be destructive to the application because files will be removed next. It could remove files that the postinstall script created. For example, a preremove script could save a specific fileset to another location before removing the rest of the filesets in the product.

  This script and the postremove script are part of the Remove phase of swremove. Within each product, preremove scripts are run (in the reverse order dictated by any prerequisites), files are removed, then all postremove scripts are run.

- **Postremove Scripts**

  This script is executed just after removing files. It is the companion script to the postinstall script. For example, if this was a patch fileset, then the preinstall script could move the original file aside, and this postremove script could move the original file back if the patch was removed.

- **Request Scripts**

  This interactive script requests a response from the user as part of software installation or configuration. Request scripts write information into a response file for later use by the configure script or other scripts. You can run requests scripts by executing the swask command or using the `ask` option with swinstall or swconfig after selection and before the analysis phase.

- **Other Scripts**

  You can include other control scripts, such as a subscript that is sourced by the above scripts. The location of the control scripts is passed to all scripts via the `SW_CONTROL_DIRECTORY` environment variable, and are denoted by the keyword `control_file` within the PSF.

Space Files

The **space** control file is not a script. It lets you define additional disk space requirements for the filesets and notes positive disk space impact on any directory or file that results from the actions of control scripts.

Each fileset or product may contain a space file. The space file lists a path and a byte size for each path. For example:

```
/tmp/space_dummy1       2000
/opt/space_dummy2       2000
/tmp/space_dummy3       3000
/mydir/                 4000
```

For each directory or file path listed in the space file, swinstall adds the size in bytes to the disk space requirements. The size reflects the maximum transient or permanent disk space required for the install.

Script Interpreter

By default, SD interprets scripts with a POSIX shell (`sh`). You can specify other script interpreters in two ways.

First, any control script can define an interpreter in the first line of the script.

Second, you can use the `interpreter` keyword to define a different interpreter for specific scripts. The syntax is:

interpreter   interpreter_name

For example:

```
control_file
   source       scripts
   tag          checkinstall
   interpreter  ksh
```

SD checks that the interpreter is available. If the interpreter is not available, the script fails. (To avoid this problem, you can use a checkinstall script to verify the existence of any script interpreters that you specify.) If SD finds the interpreter, it processes the script normally using the interpreter that you specified.

Control Script Format

A control script should be a shell script (as opposed to a binary) and written to be interpreted by the Posix.2 shell `/sbin/sh`. Korn shell (formerly `/bin/ksh`) syntax is acceptable to the Posix.2 shell. A script written for `csh` is not supported.

The script should have a simple header similar to the example below. Included in the header should also be comment lines which state the product and fileset to which the script belongs, the name of the script, the revision string as required by the *what*(1) command, and a simple copyright statement.

```
#! /sbin/sh
########
# Product: <PRODUCT>
# Fileset: <FILESET>
# configure
# @(#) $Revision: 10.30 $
########
#
# (c) Copyright MyCompany, 2001
#
########
```

General Script Guidelines

Here are some guidelines for writing control scripts:

- Consider doing most control script work within the configure script.
- All scripts are executed serially and directly impact the total time required to complete an installation, configuration, or removal task. Consider the impact control scripts will have on performance.
- The current working directory in which the agent executes a control script is not defined. Use the environment variables provided by the agent for all pathname references.
- Disk space analysis does not account for files created, copied or removed by control scripts.
- The control scripts you write may be executed several times (for example, configure, then unconfigure, then configure…) so they must be able to support multiple executions.
- You may have to re-execute or debug control scripts, especially when they generate error or warning conditions, so your scripts should be well-written and commented.
- Control script `stdout` and `stderr` are both logged, so you should restrict output to only the information the user requires.
- Make sure you specify the path to a shell that is proper for your system. If you get the following message when you execute a script:

  ```
  Cannot execute /var/adm/sw/products/PRODUCT/FILESET/
  configure. Bad file number (9).
  ```

  it means the shell in your script has a path that is not correct for your system. (HP-UX 9.X scripts = `#!/bin/sh` and HP-UX 10.X and 11.X scripts = `#!/sbin/sh`.)

Packaging Control Scripts

The following table describes the control script keywords for use in a PSF.

Table 11-2 Control Script Keywords

Keyword	Type	Size in Bytes	Example
checkinstall	path_string	1024	/mfg/sd/scripts/checkinstall
preinstall	path_string	1024	/mfg/sd/scripts/preinstall
postinstall	path_string	1024	/mfg/sd/scripts/postinstall
unpreinstall	path_string	1024	/mfg/sd/scripts/unpreinstall
unpostinstall	path_string	1024	/mfg/sd/scripts/unpostinstall
configure	path_string	1024	/mfg/sd/scripts/configure
unconfigure	path_string	1024	/mfg/sd/scripts/unconfigure
verify	path_string	1024	/mfg/sd/scripts/verify
checkremove	path_string	1024	/mfg/sd/scripts/checkremove
preremove	path_string	1024	/mfg/sd/scripts/preremove
postremove	path_string	1024	/mfg/sd/scripts/postremove
request	path_string	1024	/mfg/sd/scripts/request
control_file	path_string	1024	/mfg/sd/scripts/subscripts
fix	path_string	1024	/mfg/sd/scripts/fix
space	path_string	1024	/mfg/sd/scripts/space

The value of each keyword is the source filename for the specific control script. swpackage will copy the specified control script's filename into the depot's storage directory for the associated product or fileset, using the keyword as the tag of the stored script (for example, "configure").

You can include control script specifications or data files with the product or fileset. These are stored alongside the standard SD-UX control scripts. For example, you could specify a subscript called by the supported control scripts, or a data file read by these scripts. These additional scripts are specified using the syntax:

```
PATH[=tag]
```

If you do not specify the `tag` component, swpackage uses the *basename*(1) value of the source pathname as the tag.

Control Script Location on the File System During Execution

The checkinstall, preinstall, postinstall, and auxiliary scripts for a fileset are downloaded to a temporary directory from which they are invoked:

```
<FILESET>/control_script/var/tmp/<CATALOG_DIR>/ \
   catalog/<PRODUCT>/
```

The form of the *<CATALOG_DIR>* is: `aaaa`*<pid>*, where *<pid>* is the swinstall process ID number.

The scripts are delivered to that location from the depot immediately after Product Selection has completed, at the beginning of the Analysis phase and before any system checks have begun. The temporary directory is removed automatically upon exiting swinstall.

After successful fileset installation, all other control scripts will be located in the IPD. They will be delivered to that location from the depot as part of the installation of the fileset's other files:

```
/var/adm/sw/products/<PRODUCT>/<FILESET>/control_script
```

The location of the IPD is relative to the root directory under which the software installation is done. If the installation is to an alternate root, `/mnt/disk2` for example, then the IPD for that software will be under:

```
/mnt/disk2/var/adm/sw/products/<PRODUCT>/<FILESET>
```

NOTE All necessary directories under `/var/adm/sw` will be created by the SD-UX process. All files under those directories will be filled by SD-UX initiated processes. Files must never be delivered directly under `/var`; it is a private directory.

Using Environment Variables

All control scripts are invoked as the superuser and executed by the agent process. HP-UX provides environment variables that affect SD-UX commands and scripts. These variables fall are catgorized as follows:

- Variables that affect all SD-UX commands.
- Variables that affect all SD-UX scripts.
- Variables that affect swinstall and swremove.

Variables That Affect All SD-UX Commands

LANG

- This external variable applies to all SD commands except `install-sd`.
- Determines the language in which messages are displayed. If `LANG` is not specified or is set to the empty string, a default value of “C” is used.
- The language in which the SD agent and daemon log messages are displayed is set by the system configuration variable script, `/etc/rc.config.d/LANG`. For example, `/etc/rc.config.d/LANG` must be set to “`LANG=ja_JP.SJIS`” or “`LANG=ja_JP.eucJP`” to make the agent and daemon log messages display in Japanese.

  You may also use the `export LANG=` command.
- See the *lang(5)* man page for more information.

LC_ALL

- Determines the locale used to override any values for locale categories specified by the settings of `LANG` or any environment variables beginning with `LC_`.

LC_CTYPE

- Determines the interpretation of sequences of bytes of text data as characters (e.g., single-versus multibyte characters in values for vendor-defined attributes).

LC_MESSAGES

- Determines the language in which messages should be written.

LC_TIME

- Determines the format of dates (`create_date` and `mod_date`) when displayed by swlist. Used by all utilities when displaying dates and times in `stdout`, `stderr`, and logging.

TZ

- Determines the time zone for use when displaying dates and times.

Variables That Affect All SD-UX Scripts

SW_CATALOG

- Holds the path to the Installed Products Database (IPD), relative to the path in the `SW_ROOT_DIRECTORY` environment variable. (You can specify a path for the IPD using the `installed_software_catalog` default option.)

SW_CONTROL_DIRECTORY

- Defines the full pathname to the directory containing the script. This tells other scripts where other control scripts for the software are located (subscripts, for example).

  Also contains the response file generated by a request script. Other scripts that reference the response file access the file by referencing this variable.

  The directory is either a temporary catalog directory, or a directory within in the Installed Products Database (IPD).

Here is an example of sourcing:

```
. ${SW_CONTROL_DIRECTORY}subscript
grep something ${SW_CONTROL_DIRECTORY}datafile
```

SW_CONTROL_TAG

- Holds the tag name of the `control_file` being executed. When packaging software, you can define a physical name and path for a control file in a depot. This lets you define the `control_file` with a name other than its tag and lets you use multiple `control_file` definitions to point to the same file. A `control_file` can query the `SW_CONTROL_TAG` variable to determine which tag is being executed.

SW_LOCATION

- Defines the location of the product, which may have been changed from the default **product directory** (if the product is locatable).

  When installing to (or removing from) the primary root directory ("/"), this variable is the absolute path to the product directory. For operations on an alternate root directory, the variable must be prefixed by SW_ROOT_DIRECTORY to correctly reference product files.

  If a product is not locatable, then the value of SW_LOCATION will always be the default product directory defined when the product is packaged.

SW_PATH

- The search path for commands. A PATH variable defines the minimum set of commands available for use in a control script (for example, `/sbin:/usr/bin:/usr/ccs/sbin`).

  A control script should always set its own PATH variable, and the PATH variable must begin with $SW.PATH. The PATH should be set as follows:

  `PATH=$SW_PATH`

  `export PATH`

  Additional directories, like `/usr/local/bin`, can be appended to PATH, but you must make sure that the commands in those directories exist.

SW_ROOT_DIRECTORY

- Defines the root directory in which the session is operating, either “/” or an alternate root directory. This variable tells control scripts the root directory in which the products are installed. A script must use this directory as a prefix to SW_LOCATION to locate the product’s installed files.

  All control scripts (except for the configure and unconfigure scripts) can be executed during an install or remove task on an alternate root. If the scripts reference any product files, each reference must include the {SW_ROOT_DIRECTORY} in the file pathname.

  The scripts may only need to perform actions when installing to (removing from) the primary root directory (“/”). If so, then the SW_ROOT_DIRECTORY can be used to cause a simple exit 0 when the task is operating in an alternate root directory:

  ```
  if test "${SW_ROOT_DIRECTORY}" != "/"
  then
       exit 0
  fi
  ```

SW_SESSION_OPTIONS

- Contains the pathname of a file containing the value of every option for a particular command, including software and target selections. This lets scripts retrieve any command options and values other than the ones provided explicitly by other environment variables.

SW_SOFTWARE_SPEC

- Contains the fully qualified software specification of the current product or fileset. The software specification allows the product or fileset to be uniquely identified. (Fully qualified software specs include the `r=`, `a=`, and `v=` version components even if they contain empty strings. For installed software, `l=` must also be included.)

Variables That Affect swinstall and swremove

SW_DEFERRED_KERNBLD

- This variable is normally unset. If it is set, the actions necessary for preparing the system file `/stand/system` cannot be accomplished from within the postinstall scripts, but instead must be accomplished by the configure scripts. This occurs whenever software is installed to a directory other than `/`.
- This variable should be read only by the configure and postinstall scripts of a kernel fileset.

SW_INITIAL_INSTALL

- This variable is normally unset. If it is set, the swinstall session is being run as the back end of an initial system software installation (that is, a “cold” install).

SW_KERNEL_PATH

- The path to the kernel. The default value is `/stand/vmunix`.

SW_SESSION_IS_KERNEL

- Indicates whether a kernel build is scheduled for the current install/remove session.
- A “true” value indicates that the selected kernel fileset is scheduled for a kernel build and that changes to /stand/system are required.
- A null value indicates that a kernel build is not scheduled and that changes to /stand/system are not required.
- The value of this variable is always equal to the value of SW_SESSION_IS_REBOOT.

SW_SESSION_IS_REBOOT

- Indicates whether a reboot is scheduled for a fileset selected for removal. Because all HP-UX kernel filesets are also reboot filesets, the values of this variables is always equal to the value of SW_SESSION_IS_KERNEL.

SW_SYSTEM_FILE_PATH

- The path to the kernel’s system file. The default value is `/stand/system`.

Variables That Affect swverify

SW_IS_COMPATIBLE

- Designed to help you determine if installed software is incompatible and should be removed from a system.
- For use during the execution of a verify script, which is called by the swverify command.
- The variable will be set to true if the software being considered is compatible with the system on which it is installed.
- Set to false if the software being considered incompatible with the system on which it is installed.

Execution of Control Scripts

This section details how each control script is executed.

Details Common to All Control Scripts

- The agent runs as the superuser, therefore control scripts are always executed as the superuser. Use appropriate caution.
- Control scripts are only executed for software being installed, removed or verified in the primary root ("/") or an alternate root directory. Scripts are never executed for software in a depot.
- Each script must set its own `PATH` variable, using `SW_PATH`.
- Neither swinstall nor swremove require that the system be shut down. Control scripts must work correctly on both quiet single-user systems and active multi-user systems. They must deal properly with unremovable running programs. They might have to shut down or start up processes that they own themselves to succeed.
- Control scripts can be re-executed. If a script is run more than once, it should produce the same results each time. The second execution should not produce any error messages or leave the system in a state different than before it was run.

  A script should be executable after its fileset was loaded without damaging the new fileset with which it is associated.

  For example, if you must copy a file from under `/usr/newconfig` to another location, use the `cpio -p` command to copy it rather than the cp command to move it, or check for the absence of the `/usr/newconfig` version before attempting the move. (The *cpio*(1) command may be preferred over *cp*(1) because cpio copies the mode, owner, and group permissions.)
- Control scripts must exit with a return value of zero (`exit 0`) if no serious errors occur (no error or warning messages printed, as described in the "Control Script Input and Output" on page 397.) They must return 1 (`exit 1`) in case of any serious errors, and 2 (`exit 2`) for warnings.

  All messages produced by control scripts are redirected to the agent logfile.

- The set of control scripts executed during a particular phase of a task are always executed in prerequisite order the scripts of each prerequisite product/fileset are executed before the script of the dependent fileset.
- All control scripts are readable by any other control script.

Checkinstall Scripts

- Checkinstall scripts are executed during the Analysis phase of a swinstall session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}checkinstall`
- A checkinstall script must not modify the system.
- A checkinstall script determines whether the product/fileset can be installed by performing checks beyond those performed by swinstall. Example checks include checking to see if the product/fileset is actively in use, or checking that the system run-level is appropriate.
- If you are using a request script as part of the install, the checkinstall script should:
  - Verify that the response file exists.
  - Prevent swinstall from “hanging” if:
    - A script tries to read a response file that does not exist, or
    - The install or configuration relies on information in the missing response file.
- If the checkinstall script fails, the fileset will not be installed. The interactive interface of swinstall will notify you that the checkinstall script has failed. Then you can: diagnose the problem, fix it and re-execute the analysis phase; or unselect the product/fileset. The non-interactive interface tells you about each individual checkinstall failure and the filesets are not installed.
- A checkinstall script is executed for installations into the primary root (“/”) or an alternate root. Since most of the actions of this script will involve checking the current conditions of a running system (that is, the primary root), it may not need to perform any actions when the product/fileset is being installed into an alternate root.

Preinstall Scripts

- Preinstall scripts are executed during the Load phase of a swinstall session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}preinstall`

- The preinstall script for a product is executed immediately before the fileset's files are installed.
- A preinstall script should perform specific tasks preparatory to the files being installed. The swinstall session will proceed with installing the files regardless of the return value from a preinstall script. Example actions include removing obsolete files (in an update scenario).
- A preinstall script is executed for installations into the primary root ("/") or an alternate root. The scope of actions of a preinstall script should be within the product itself (that is, the files within the product's directory).

Postinstall Scripts

- Postinstall scripts are executed during the Load phase of a swinstall session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}postinstall`

- The postinstall script for a product is executed immediately after the fileset's files are installed.
- A postinstall script should perform specific tasks related to the files just installed. The swinstall session will proceed with the remainder of the session (for example, configuration) regardless of the return value from a postinstall script. Example actions include adding a kernel driver to the system file or moving a file from under `/usr/newconfig` to its correct place in the file system.
- A postinstall script is executed for installations into the primary root ("/") or an alternate root. The scope of actions of a postinstall script should be within the product itself (that is, the files within the product's directory).
- The customization or configuration tasks that must be performed to enable the product/fileset for general use should not be done in the postinstall script, but the configure script (described below).

Configure Scripts

- Configure scripts are executed during the Configuration phase of a swinstall session. SD expects configure scripts at system start-up if the swinstall session triggers a system reboot. The swconfig command can also execute configure scripts. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}configure`

- A configure script is only executed for installations into the primary root ("/"). If you choose to defer configuration in the swinstall session, then the configure script will be executed by a swconfig session at some time after the installation completes.
- A configure script is usually executed only when the product/fileset is in the installed state.
- A configure script is the primary way to move a product/fileset from the installed state to the configured state. The script should perform all (or most of) the activities needed to enable the product/fileset for use.
- A configure script can use configuration information provided by the user and collected by a request script.
- When an existing version of a product is updated to a new version, the configure script(s) for the new version must perform any unconfigurations-configurations of the old version that are necessary to properly configure the new version. The unconfigure script(s) for the old version are not executed.
- Configure scripts are for architecture-dependent actions because they will always be run on the architecture of the install target.
- Configure scripts are the best place for removing files and updating the IPD, since the system is not in transition (i.e. as in an update).
- A configure script can help with software updates as well as new installs. The script must also be able to handle reinstallation and should include appropriate error control if data destruction is possible.

Unconfigure Scripts

- Unconfigure scripts are executed during the Unconfiguration-Configuration phase of a `swremove` session. They can also be executed by the swconfig command. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}unconfigure`

- An unconfigure script is executed only for software installed into the primary root ("/").
- An unconfigure script is re-executed even when the product/fileset is in the configured state.
- An unconfigure script is the primary way to move a product/fileset from the configured state back to the installed state. The script should perform all (or most of) the activities needed to disable the product/fileset for use.
- An unconfigure script must undo all configuration tasks performed by its companion configure script. The user should be able to configure, unconfigure, configure, etc. an installed product/fileset and always end up with the same configured result.

Verify Scripts

- Verify scripts are executed by the swverify command. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}verify`

- A verify script must not modify the system.
- A verify script is the primary way to check the configuration tasks performed by a configure script for correctness and completeness.
- A verify script is executed for installations into the primary root ("/") or an alternate root. Since most of the actions of this script will involve checking the current conditions of a configured product/fileset (in the primary root), it may not need to perform any actions for a product/fileset installed into an alternate root directory.
- An environment variable, SW_IS_COMPATIBLE, can help a verify script determine if installed software is compatible with the system on which it is installed. See "SW_IS_COMPATIBLE" on page 386.

Fix Scripts

- Fix scripts are executed by the swverify command. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}fix`

- A fix script can be used to correct attribute problems detected by a verify script.
- A fix script can create missing directories, correct file modifications (mode, owner, group, major, and minor), and recreate symbolic links.

Checkremove Scripts

- Checkremove scripts are executed during the Analysis phase of a swremove session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}checkremove`

- A checkremove script must not modify the system.
- A checkremove script determines whether the product/fileset can be removed by performing checks beyond those performed by swremove. Example checks include checking to see if the product/fileset is actively in use.
- If the checkremove script fails, no filesets in the product will be removed. The GUI/TUI interface of swremove notifies you that the checkremove script has failed. You can then: diagnose the problem, fix it, and re-execute the analysis phase; unselect the target system(s) in question; or unselect the product/fileset. The command line interface notifies you for each individual checkremove failure, and no filesets in that product are removed.
- A checkremove script is executed for installations into the primary root (“/”) or an alternate root. Since most of the actions of this script will involve checking the current conditions of a running system (that is, the primary root), it may not need to perform any actions when the product/fileset is being removed from an alternate root.

Preremove Scripts

- Preremove scripts are executed during the Remove phase of a swremove session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}preremove`

- All preremove scripts for a product are executed immediately before the product's files are removed.
- A preremove script should perform specific tasks preparatory to the files being removed. The swremove session will proceed with removing the files regardless of the return value from a preremove script. Example actions include removing files created in the postinstall script.
- A preremove script is executed for installations into the primary root ("/") or an alternate root. The scope of actions of a preremove script should be within the product itself (that is, the files within the product's directory).
- The de-customization or unconfiguration-configuration tasks which must be performed to disable the product/fileset for general use must not be done in a preremove script, instead they should be done in an unconfigure script (described above).

Postremove Scripts

- Postremove scripts are executed during the remove phase of a swremove session. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}postremove`

- All postremove scripts for a product are executed immediately after the product's fileset files are removed.
- A postremove script should perform specific tasks related to the files just removed. The swremove session will proceed with the remainder of the session regardless of the return value from a postremove script. Example actions include:
  - — Removing any files still remaining after preremove and the swremove file removal have completed.
  - — Removal of directories wholly owned by the fileset and which have been emptied by the file removal.

- A postremove script is executed for installations into the primary root ("/") and an alternate root. The scope of actions of a postremove script should be within the product itself (that is, the files within the product's directory).
- The de-customization or unconfiguration-configuration tasks which must be performed to disable the product/fileset for general use should not be done in the postremove script, instead they should be done in the unconfigure script (described above).

Request Scripts

- Request scripts are interactive scripts that request a response from the user as part of software installation or configuration. The pathname of the script being executed is:

  `${SW_CONTROL_DIRECTORY}request`
- Request scripts write information into a response file for later use by the configure script or other scripts. You can run requests scripts by executing the swask command or using the `ask` option with swinstall or swconfig after selection and before the analysis phase.
- The POSIX default for request scripts is a shell script. The shell script must be able to:
  - — Ask questions of the user.
  - — Read the user's answer.
  - — List all current user responses in a redrawn screen.
  - — Ask the user to confirm an answer and continue or to go back.
- The request script stores the user response in a response file. The path of the response file is accessible by the SW_CONTROL_DIRECTORY environment variable.
- The POSIX recommendation for response file format is the SVR4 model of attribute/value pairs. Answers should be written to the response file in `env_var=`*`value`* format so that the response files can be easily used by other control scripts.
- When you use a request script to get install information, HP recommends that you use a checkinstall script to check for proper execution of the request script. The checkinstall script should:

- Verify that the response file exists.
- Prevent swinstall from “hanging” if:
  - A script tries to read a response file that does not exist.
  - The install or configuration relies on information in the missing response file.

Execution of Other Commands by Control Scripts

Every command executed by a control script is a potential source of failure because the command may not exist on the target system. Your script can use any command conditionally, if it checks first for its existence and executability, and if it does not fail when the command is unavailable.

- If the target system(s) conform with the POSIX 1003.2 Shells and Utilities standard, then the Execution Environment Utilities of this standard will also be available.
- If a fileset has a prerequisite dependency on another product/fileset, then most of the control scripts for the dependent fileset can use the commands of the required product/fileset, if the $ROOT_DIRECTORY is `/`. (All commands perform their tasks in prerequisite order).
- Commands should be referenced relative to the path components specified in the `PATH` variable. (See the discussion of `PATH` and the `SW_PATH` environment variable above.)

Control Script Input and Output

- Except for request scripts, control scripts must not be interactive. This includes messages such as, `Press return to continue`.
- Except for request scripts, all control scripts are executed by the agent on the target systems. Request scripts are executed by the controller (swinstall, swconfig, or swask).
- Except for request scripts, no method of input to control scripts is supported. Request script data is input by the user through the swask command or the `ask` option for swinstall or swconfig.
- Control scripts must write messages for error and warning conditions to `stderr` (`echo &>2`), and write all other messages to stdout. Control scripts must not write directly to `/dev/console` or attempt any other method of writing directly to the display.

  The `stdout` and `stderr` from a control script is redirected by the agent to the log file (`var/adm/sw/swagent.log`) within the primary or alternate root directory in which the task is being performed.

  For interactive swinstall and swremove sessions, you can display and browse this logfile.
- Only minimal, essential information should be emitted by control scripts. Ideally, no output is emitted if the script successfully performs all of its actions.
- In the agent logfile, the execution of each control script is prefaced by a “begin execution” message:

  ```
  * Running "checkinstall" script for product "PRODUCT"
  * Running "checkinstall" script for fileset
    "PRODUCT.FILESET".
  ```

  Any messages generated by the script will follow. If the script returns a value other than 0 (SUCCESS), then a concluding message such as the following, is written:

```
ERROR: The "unconfigure" script for "PRODUCT.FILESET"
failed (exit code "1"). The script location was
"/var/adm/sw/products/PRODUCT/FILESET/unconfigure".
* This script had errors but the execution of this
product will  still  proceed. Check the above output
from the script for further details.

WARNING: The "unconfigure" script for
"PRODUCT.FILESET" failed (exit code "2"). The script loc
ation was
"/var/adm/sw/products/PRODUCT/FILESET/unconfigure"
* This script had warnings but the execution of this
product will still proceed.   Check the above output
from the script for further details.
```

- The messages written by a control script must conform to the following format conventions whenever possible.

  1. Never emit blank lines.
  2. All output lines must have one of these forms:

     ```
      ERROR:     text
      WARNING:   text
      NOTE:      text
     blank      text
     ```

     In each case, the keyword must begin in column 1, and the *text* must begin in column 10 (indented nine blanks).
  3. Choose the keyword (ERROR, WARNING, NOTE, or blank) as follows:

ERROR:	Cannot proceed, may need corrective action.
WARNING:	Can proceed, but something went wrong and may need action.
NOTE:	Can proceed, but something happened that is out of the ordinary or worth special attention. (Not just a status message.)
blank	Generic progress and status messages (keep them to a necessary minimum).

     Do not start a line with an asterisk (*) character. This is reserved for operational messages printed by the agent so you can easily distinguish them from other messages.

4. If the message text requires more than a single 72-character line, break it into several 72-character lines. Indent all lines after the first. For example:

   ```
   NOTE: To install your new graphics package, you
   must turn on the lights in the next room.
   Please turn them off when you leave.
   ```

5. Do not use tab characters in any messages.

- Scripts execute other commands which may unexpectedly fail and emit output not in the above format. Wherever you suspect a failure is possible or likely (and it is reasonable to do so) redirect the standard output or error of the executed command to `/dev/null` or to a temporary file. Then emit a proper-format message based on the return code or on output from the command. For example:

  ```
  /bin/grep bletch /etc/bagel 2c&>/dev/null
  if[$?=1]
  then
     echo “ERROR: Cannot find bletch in /etc/bagel.” |&>2
  fi
  ```

- Follow these conventions to ensure a control script’s messages have a similar look and feel to the messages generated by the agent (and the commands themselves).

  — Use full sentences wherever possible. Avoid terseness.

  — Start sentences and phrases with capital letter and end with period.

  — Put two blanks after period; one after colons, semicolons, and commas.

  — Use uppercase first letters of phrases after colons. (This helps break up the message into digestible “bites” of information.)

  — Surround product, fileset, directory, and file names, and other variable-valued strings with quotes. For example:

  ```
  echo "ERROR:  Cannot open file \"$file\"." &>2
  ```

  — Write in the present tense. Avoid “would”, “will”, and similar verb tenses. Also avoid past tense except where necessary.

  — Use “cannot” rather than “can’t”, “could not”, “couldn’t”, “unable to”, “failed to”, and similar phrases.

— Write messages that make sense to system administrators and users. Consider your audience.

File Management by Control Scripts

- All files created by a preinstall, postinstall, or configure script must be removed by a companion postremove, preremove or unconfigure script.

  Files created by scripts are not known by the swremove command, and will not get removed when it removes those files installed by swinstall. If you want script-created files removed by SD, you will have to add them to the IPD by either the swmodify command or the `control_utils` function `IPD_addfile`.
- If any files in the previous revision of a product have changed names or became obsolete, a product/fileset preinstall or postinstall script in the new revision of the product must remove the old files. The agent does not remove the files in an existing product/fileset before updating it to a newer revision.

NOTE It is necessary to perform the cleanup task of any previous revision that can be updated to the new revision. Sometimes this is more than just the previous revision.

Testing Control Scripts

The following testing suggestions do not cover all test scenarios. There may still be problems with a control script even after doing this testing. For example, you may test installing/removing individual filesets. But there might be some interactions that are discovered only after all the filesets are installed on or removed from the system.

Similarly, you may test the control scripts on a fully loaded system and miss a problem when you execute a command in your script that is not part of the base (or core) system. If your target system does not contain the particular command, your script may fail.

Testing Installation Scripts

For checkinstall, preinstall, and postinstall scripts you should perform at least these tests. All tests can be performed on the local system (that is, by doing local installs).

1. The basic test:
   - Run swinstall to install the full product (that is, all the filesets). To avoid testing the configure script(s), either do not include any in the product, or set the `defer_configure` option to “true.”
   - After the installation completes, check the `<${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the scripts or the format/contents of the messages generated by the scripts.
   - Study the resulting file system to see if the scripts performed the expected actions.
   - Re-run the test by re-installing the same product.
2. If you want to avoid the time spent loading files, then set the `reinstall_files` option to “false” and the `reinstall_files_use_cksum` option to “false.”
3. If a previous version of the product can be updated to this version, then re-run the test by updating this product where the previous version has been installed.

4. If your checkinstall script can generate error or warning conditions based on the current activity or configuration of the target system, then enable those conditions to ensure that the checkinstall script correctly detects them.
5. Re-run the test by installing into an alternate root directory (`swinstall -r`) instead of the primary root directory (“/”). Make sure that the scripts perform all of their operations (if any) within the alternate root directory. (This verifies the correct use of ${SW_ROOT_DIRECTORY} by your scripts.)
6. If your product is locatable (that is, it can be installed into a different location), then re-run the tests by installing the product into a different location (swinstall *`product`*`:new_location`). Make sure that the scripts perform all of their operations in the new location, and not the default location. (This verifies the correct use of $SW_LOCATION by your scripts.)
7. If you have a complex script, run additional tests for your product that you feel will give you confidence your product has been installed correctly on the system. For example, only install certain subsets of your product instead of the full product.

Testing Configuration Scripts

For configure, verify, and unconfigure scripts you should perform at least these tests. All tests can be performed on the local system (that is, by doing local installs).

1. Run swinstall to install the full product (that is, all the filesets). Let the installation process perform the configuration task (and run your configure script(s)).
   - After the installation and configuration completes, check the `${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the configure script or the format/contents of the messages generated by it.
   - Study the resulting file system to see if the configure script performed the expected actions.
   - Test the product itself to see if the necessary configuration tasks were performed such that the product is ready to use.

2. Run swremove to remove the configured product.
   - After the unconfiguration and removal completes, check the `${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the unconfigure script or the format/contents of the messages generated by it.
   - Study the resulting file system to see if the unconfigure script performed the expected “undo” actions.
3. Run swinstall to install the full product again. Set the `defer_configure` option to “false” to avoid executing the configure scripts.
   - After the installation completes, run swconfig to configure your product.
   - Study the resulting file system to see if the configure script performed the expected actions.
   - Test the product itself to see if the necessary configuration tasks were performed such that the product is ready to use.
   - Now run `swconfig -u` to unconfigure your product.
   - Study the resulting file system to see if the unconfigure script performed the expected “undo” actions.
   - Run swconfig again to re-configure your product.
   - Study the resulting file system to see if the configure script performed the expected actions.
4. Run swverify to execute the verify script(s).
   - After the verification completes, check the `${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the verify script or the format/contents of the messages generated by it.
5. If a previous version of the product can be updated to this version, then re-run the first test by updating this product to a system where the previous version has been installed and configured.
6. Note that configure and unconfigure scripts are never run unless the `${SW_ROOT_DIRECTORY}` is `/`. However, verify scripts are run in both cases.

7. If your product is locatable (that is, it can be installed into a different location), then re-run the tests by installing and configuring the product in a different location. Make sure that the scripts perform all their operations in the new location, and not the default location. (This verifies the correct use of `$SW_LOCATION` by your scripts.)
8. If you have a complex script, run additional tests for your product that you feel will give you confidence your product has been installed correctly on the system. For example, only install certain subsets of your product instead of the full product.

Testing Removal Scripts

For checkremove, preremove, and postremove scripts you should perform at least these tests. All tests can be performed on the local system (that is, by doing local installs). There is no value gained by testing your scripts by installing to remote target systems.

1. Run swinstall to install the full product (that is, all the filesets). Avoid configuration by setting the `defer_configure` option to false.
   - Run swremove to removed the unconfigured product.
   - After the removal completes, check the `${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the removal scripts or the format/contents of the messages generated by the scripts.
   - Study the resulting file system to see if the removal scripts performed the expected actions.
2. Run swinstall to install the full product (that is, all of the filesets). Let the installation process perform the configuration task (and run your configure script(s)).
   - Run swremove to removed the configured product.
   - After the unconfiguration and removal completes, check the `${SW_ROOT_DIRECTORY}var/adm/sw/swagent.log` file for any problems, either in the removal scripts or the format/contents of the messages generated by the scripts.
   - Study the resulting file system to see if the removal scripts performed the expected actions.

3. If your checkremove script can generate error or warning conditions based on the current activity or configuration of the target system, then enable those conditions to ensure that the checkremove script correctly detects them.
4. Re-run the first test by installing into an alternate root directory (`swinstall -r`) instead of the primary root directory ("/"). Make sure that the scripts perform all of their operations (if any) within the alternate root directory. (This verifies the correct use of `${SW_ROOT_DIRECTORY}` by your scripts.)
5. If your product is locatable (that is, it can be installed into a different location), then re-run the tests by installing the product into a different location. When removing the product, make sure that the removal scripts perform all of their operations in the new location, and not the default location. (This verifies the correct use of `$SW_LOCATION` by your scripts.)
6. If you have a complex script, run additional tests for your product that you feel will give you confidence your product has been installed correctly on the system. For example, only install certain subsets of your product instead of the full product, then perform the remove operations. (Or only remove subsets of the fully installed product.)

Requesting User Responses (swask)

SD-UX packaged applications can use interactive control scripts to query a user and obtain installation or configuration information that cannot be known at package time. For example, different hardware or OS versions may require different configuration, or some software may need a specific IP address or hostname for configuration.

SD-UX runs the interactive control scripts by the swask command or by the `ask` default option for the swinstall and swconfig commands. (SD-UX does not query the user but the control script does.)

Using swask

- The swask command runs interactive software request scripts for the software objects selected.
- These scripts store the responses in a response file (named `response`) for later use by the swinstall or swconfig commands. (swinstall and swconfig can also run the interactive request scripts directly, using the `ask` option.)
- A response file is generated for each piece of selected software that has a corresponding request script.
- swask uses the command-line only; there is no Graphical User Interface.

Syntax

`swask [-v] [-c` *`catalog`*`] [-C` *`session_file`*`] [-f` *`software_file`*`] [-s` *`source`*`][-S` *`session_file`*`][-x` option=value] [`-X` *`options_file`*`] [`*`software_selections`*`][@`*`target_selections`*`]`

Options and Operands

-v Turns on verbose output to `stdout` and displays all activity to the screen.

-c *`catalog`* Specifies the pathname of an exported catalog which stores the response files created by the request script. swask creates the catalog if it does not already exist.

If the `-c` *`catalog`* option is omitted and the source is local, swask copies the response files into the source depot: *`distribution`*`.path/`*`catalog`*.

`-C` *`session_file`*
: Run the command and save the current option and operand values to a session_file for re-use in another session. See “Session Files” on page 61.

`-f` *`software_file`*
: Read a list of software selections from a separate file instead of (or in addition to) the command line. See “Software Files” on page 58.

`-S` *`session_file`*
: Run the command based on values saved from a previous installation session, as defined in *`session_file`*. See “Session Files” on page 61.

`-s` *`source`*
: Use the software source specified by *source* instead of the default, `/var/spool/sw`. The syntax is:

[*`host`*]`[:]`[*`/directory`*]

host may be a host name, domain name, or internet address (for example, `15.1.48.23`). *directory* is an absolute path.

`-X` *`option_file`*
: Read session options and behaviors from *`option_file`*. See “Changing Command Options” on page 409.

`-X` *`option_file`*
: Read a list of options and behaviors from *`option_file`*.

`software_selections`
: The software objects for which the request script will be executed. See “Software Selections” on page 56.

`target_selections`
: The target of the command. See “Target Selections” on page 58.

Changing Command Options

You can change the behavior of this command by specifying additional command-line options when you invoke the command (using the `-x` option) or by reading predefined values from a file. The following table shows the options and default values that apply to swconfig.

Table 11-3 swask Command Options and Default Values

• admin_directory=/var/adm/sw	• logdetail=false
• ask=true	• logfile=/var/adm/sw/swask.log
• autoselect_dependencies=true	• loglevel=1
• autoselect_patches=true	• patch_filter=*.*
• enforce_scripts=true	• run_as_superuser=true
• installed_software_catalog=products	• verbose=1
• log_msgid=0	

For More Information

See Appendix A, “Command Options,” on page 421 for complete descriptions of each default.

Request Script Tasks and Examples

You can run request scripts from the swinstall or swconfig commands by setting the ask option to true. This tells the commands to run request scripts (if any exist) in addition to performing install or configuration tasks. (Note that the value of the ask option if false for both swinstall and swconfig but is true for swask.)

swask Examples

Run all request scripts from the default depot (`/var/spool/sw`) and write the response file (`response`) back to the same depot:

```
swask -s /var/spool/sw \*
```

Run the request script for `Product1` from depot `/tmp/sample.depot.1` on remote host `swposix`, create the catalog `/tmp/test1.depot` on the local controller machine, and place the response file (`response`) in the catalog:

```
swask -s swposix:/tmp/sample.depot.1 \
     -c /tmp/test1.depot Product1
```

Run request scripts from remote depot `/tmp/sample.depot.1` on host `swposix` only when a response file is absent, create the catalog `/tmp/test1.depot` on the local controller machine, and place the response file (`response`) in the catalog:

```
swask -s swposix:/tmp/sample.depot.1 \
      -c /tmp/test1.depot -x ask=as_needed \*
```

swinstall Examples

To install all the software from local depot `tmp/sample.depot.1` using any response files generated by request scripts:

```
swinstall -s /tmp/sample.depot.1 -x ask=true \*
```

To install `Product1` from remote depot `/tmp/sample.depot.1` on host `swposix` and use an existing response file (previously generated by the swask command) located in `/tmp/bar.depot`:

```
swinstall -s swposix:/tmp/sample.depot.1 \
        -c /tmp/bar.depot Product1
```

To install all products in remote depot `/tmp/sample.depot.1` on host `swposix`, use any response files generated by request scripts, create catalog `/tmp/bar.depot` and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 \
         -c /tmp/bar.depot -x ask=true \*
```

To install all products in remote depot `/tmp/sample.depot.1` on host `swposix`, use response files, run request scripts only when a response file is absent, create catalog `/tmp/bar.depot` and copy all response files to the new catalog:

```
swinstall -s swposix:/tmp/sample.depot.1 \
         -c swposix:/tmp/bar.depot -x ask=as_needed \*
```

swconfig Examples

To configure `Product1`, use any associated response files generated by a request script, and save response files under `/tmp/resp1`:

```
swconfig -x ask=true -c /tmp/resp1 Product1
```

12 Nonprivileged SD

This chapter provides general guidelines on how to set up Software Distributor to run in nonprivileged mode.

Table 12-1 Chapter Topics

Topics:
“Overview” on page 414
“Setting Up Nonprivileged Mode” on page 416
“Default Configuration” on page 418
“Alternative Configuration” on page 419

Overview

The nonprivileged mode of SD-UX lets users access application software based on their file system permissions rather than super-user privilege implemented by SD-UX ACLs. Nonprivileged mode is honored by almost all SD commands. You can use nonprivileged mode for all aspects of developing, distributing, and managing applications.

Who Can Benefit?

Nonprivileged SD-UX is primarily intended for administrators of large data centers who must manage in-house applications without using super-user privilege. You might not benefit from this feature if you are a casual user wanting to manage your own applications—unless you are experienced enough at packaging software to take advantage of nonprivileged mode.

How Does It Work?

In nonprivileged mode, most SD-UX operations are done according to the invoking user's uid, gid, and umask. In this mode, logfiles and the installed software catalog usually found in `/var/adm/sw` are stored by default in user-specific admin directories at `/var/home/USER_NAME/sw` (in which `USER_NAME` is the user's log-in name). Location of the user's admin directory and installed software catalog can be customized using default options.

While you are using nonprivileged mode, you can also package and copy applications that won't be used for nonprivileged mode. However, you must use the normal mode of SD-UX (that is with `run_as_superuser` set to true and permissions granted by ACLs) to install such applications.

When packaging, file system access on the install target must be considered. See "Packaging Software for Use in Nonprivileged Mode" on page 416.

Limitations

- Remote targets are not allowed with SD-UX remote operations, except for swlist access to remote systems and commands that can normally access remote depots. Access to such remote systems is determined by the SD ACLs on the remote system.
- Nonprivileged mode cannot be used to manage HP-UX operating system software or patches to it.
- A swinstall or swcopy in nonprivileged mode cannot read a source depot on a local writable file system that was created with super-user privileges (that is, created by a super-user, or created by a non super-user when the `run_as_superuser` option is set to true and using ACL permissions). This limitation does not apply to tape or CD-ROM source depots.
- Swinstall and swcopy in nonprivileged mode can read any remote source depot as allowed by ACLs, can read local source depots created by the invoking user in nonprivileged mode, and (depending on the umask of other users) can read local source depots created by other users in nonprivileged mode.

Setting Up Nonprivileged Mode

Nonprivileged SD is controlled by two options:

- `admin_directory`
- `run_as_superuser`

The `run_as_superuser` option turns nonprivileged mode on or off and is all that is necessary to run the default configuration. (See “Turning On Nonprivileged Mode” on page 417 and “Default Configuration” on page 418.)

The `admin_directory` option lets you set up an alternative configuration. (See “Alternative Configuration” on page 419.)

Packaging Software for Use in Nonprivileged Mode

In addition to these options, software applications to be used under nonprivileged mode have special packaging requirements.

For nonprivileged mode to function:

- You must package applications and install them so that the files are installed in locations writable by the user who will install the applications. This can be done by:
  - Using the directory keyword in the PSF during packaging
  - By appending a `location` to the software specifications when you invoke a command from the command line. (See “Software Selections” on page 56.)
- Scripts packaged into the application must be designed not to require super-user privilege.

Turning On Nonprivileged Mode

SD functions in nonprivileged mode only when the `run_as_superuser` option is set to false and the invoking user is not super-user.

This option applies to all SD-UX commands except swagent, swagentd, swjob, and install-sd. When you set this option to false, any command to which it applies will run in nonprivileged mode. For example:

- Including **`-x run_as_superuser=false`** on the command line invokes nonprivileged mode for that command only.
- Including **`-x run_as_superuser=false`** in your `$HOME/.swdefaults` directory invokes nonprivileged mode for any or all SD-UX commands that you run.
- Including **`-x run_as_superuser=false`** in `/var/adm/sw/defaults` invokes nonprivileged mode for all SD-UX commands on the system.

See Appendix A, “Command Options,” on page 421 for complete information on using these options.

NOTE This option is ignored (treated as true) when the invoking user is super-user.

How Nonprivileged Mode Changes SD-UX Behavior

When the `run_as_superuser` option is set to the default value of true, SD-UX operations are performed normally, with permissions for operations either granted to a local super-user or set by SD ACLs. (See Chapter 9, “SD-UX Security,” on page 255 for details on ACLs.)

When `run_as_superuser` is set to false and the invoking user is local and is *not* super-user, nonprivileged mode is invoked:

- Permissions for operations are based on the user’s file system permissions.
- SD ACLs are ignored.
- Files created by SD have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user’s umask.

Default Configuration

The default configuration of nonprivileged mode is to have a central location for user-installed software catalogs.

When the `run_as_superuser` option is false and the `admin_directory` option is not set, SD-UX logfiles and installed software catalogs are stored in user-specific directories at `/var/home/USER_NAME/sw` (where `USER_NAME` is replaced by the invoking user name).

Putting logfiles and installed software catalog in a central location avoids problems when users install software on the system outside of their home directories and user home directories are NFS mounted across many systems.

You can enable nonprivileged mode for all users by setting the `run_as_superuser` option to false in `/var/adm/sw/defaults`.

Individual users can override the default chosen by the system administrator, by setting the `run_as_superuser` option to true or false in their `$HOME/.swdefaults` file or on the command line.

Alternative Configuration

An alternative configuration of nonprivileged mode sets up user-installed software catalogs in each user's home directory. You can use the `admin_directory` option in `/var/adm/sw/defaults` to indicate a path beginning with `HOME` or `/HOME`, so that the default administration directory used by SD-UX during nonprivileged mode is in each user's home directory. (A value of `HOME/.sw` works well for this purpose.)

Individual users can override this in their `$HOME/.swdefaults` file or on the command line.

Setting the Admin Directory Option

This option lets you specify the location for logfiles and the default parent directory for the installed software catalog. Values are as follows:

`admin_directory=/var/adm/sw` (for normal mode)

`admin_directory=/var/home/LOGNAME/sw` (for nonprivileged mode)

The default value is `/var/adm/sw` for normal operations. For nonprivileged mode (that is, when the `run_as_superuser` option is set to true):

- The default value is forced to `/var/home/LOGNAME/sw`.
- The path element `LOGNAME` is replaced with the name of the invoking user, which SD-UX reads from the system password file.
- If you set the value of this option to `HOME/path`, SD-UX replaces `HOME` with the invoking user's home directory (from the system password file) and resolves path relative to that directory.

  For example, if you specified `HOME/my_admin` for this options, the location would resolve to the `my_admin` directory in your home directory.

This option applies to swinstall, swcopy, swremove, swconfig, swverify, swlist, swreg, swacl, swpackage, swmodify.

A Command Options

This appendix reviews the basics of altering SD-UX command options and provides an alphabetic list of all options and their default values.

Table A-1 Chapter Topics

Topics:
"Changing Command Options" on page 422
"Options Listed Alphabetically" on page 424

Changing Command Options

Changing the option values lets you change command behavior and tailor SD-UX policies to your needs. You can change options using predefined files, values you specify directly on the command-line, or the GUI Options Editor from the Options menu. Altering option values using files can help when you don't want to specify command behavior every time you invoke the command.

These rules govern the way the options work:

- Option values specified in `/var/adm/sw/defaults` affect all SD-UX commands on that system. This file can change options for all commands to which an option applies or for specific commands only.
- Option values in your personal `$HOME/.swdefaults` file affect only you and not the entire system.
- Option values read from a session file affect only that session.
- Options changed on the command line by the `-X` *`option_file`* or the `-x` *`option=value`* arguments override the system-wide and personal defaults files but affect only that invocation of the command.

For system-wide policy setting, use the `/var/adm/sw/defaults` files. Keep in mind, however, that users may override these values with their own `$HOME/.swdefaults` file, session files, or command line changes.

The template file `/usr/lib/sw/sys.defaults` provides documentation for all options, and contains instructions for an easy way to change system-wide or personal default files.

The template file documents as comments all SD-UX command options, the commands to which they apply, their possible values, and the resulting system behavior. You can copy values from this file into the system defaults file (`/var/adm/sw/defaults`), your personal defaults file (`$HOME/.swdefaults`), or an input file and uncomment them to affect your system behavior.

Option files use the syntax:

```
[command.]option=value
```

- The optional *`command`* is the name of a SD-UX command. Specifying a command name changes the default behavior for that command only. A period must follow a command name.
- *`option`* is the name of the default option. An equals sign must follow the option name.
- *`value`* is one of the allowable values for that option.

NOTE Use caution when changing default option values. They allow useful flexibility but can produce harmful results if changed to a value that is inappropriate for your needs.

NOTE Options in the defaults file are read as part of command initialization. Because the daemon is already running, you must restart the daemon after changing daemon options for the system to recognize those options. To restart the daemon, type:

```
/usr/sbin/swagentd -r
```

See Also “Using Command Options” on page 59 for examples.

Options Listed Alphabetically

- `admin_directory=/var/adm/sw` (for normal mode)
  `admin_directory=/var/home/LOGNAME/sw`
  (for nonprivileged mode)

  The location for logfiles and the default parent directory for the installed software catalog. The default value is `/var/adm/sw` for normal operations.

  For nonprivileged mode (that is, when the `run_as_superuser` default option is set to true):

  — The default value is `/var/home/LOGNAME/sw`

  — The path element LOGNAME is replaced with the name of the invoking user, which SD-UX reads from the system password file.

  — If you set the value of this option to `HOME/path`, SD-UX replaces HOME with the invoking user's home directory (from the system password file) and resolves path relative to that directory. For example, `HOME/my_admin` resolves to the `my_admin` directory in your home directory.

  — If you set the value of the `installed_software_catalog` option to a relative path, that path is resolved relative to the value of this option.

  Nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. For a full explanation of nonprivileged SD-UX, see Chapter 12, “Nonprivileged SD,” on page 413.

  See also the `installed_software_catalog` and `run_as_superuser` options.

  Applies to all commands except swagent, swagentd, and install-sd.

- **`agent=/usr/lbin/swagent`**

  This is the default location of the executable invoked to perform agent tasks.

  Applies to swagentd.

- **`agent_auto_exit=true`**

  Causes the target agent to automatically exit after execute phase, or after a failed analysis phase. This is forced to false when the controller is using an interactive UI, or when `-p` (preview) is used.

  Enhances network reliability and performance.

  The default is true. The target agent automatically exits when appropriate.

  If set to false, the target agent does not exit until the controller explicitly ends the session.

  Applies to swconfig, swcopy, swinstall, swremove, and swverify.

- **`agent_timeout_minutes=10000`**

  Causes a target agent to exit if it has been inactive for the specified time.

  You can use this default value to make target agents more quickly detect lost network connections. RPC typically detects lost connections very quickly, but it can take as long 130 minutes to detect a lost connection. The recommended value is the longest period of inactivity expected in your environment.

  For command line invocation, a value between 10 and 60 minutes is suitable. More than 60 minutes is recommended when you use the GUI.

  The default is 10,000 minutes, slightly less than seven days.

  Applies to swcopy, swinstall, swlist, swremove, and swverify.

- **`allow_downdate=false`**

  Normally set to false, so installing an older version of software than already exists is disallowed. This keeps you from installing older versions by mistake. Additionally, many software products do not support this “downdating.”

  If set to true, a previous version can be installed but SD-UX issues a warning message.

  Applies to swinstall.

- **`allow_incompatible=false`**

  Normally set to false, only software compatible with the local host is allowed to be installed or configured.

  If set to true, no compatibility checks are made.

  Applies to swconfig, swinstall, and swverify.

- **`allow_multiple_versions=false`**

  Normally set to false, so installed or configured multiple versions (for example, the same product, but a different revision, installed into a different location) are disallowed. Even though multiple *installed* versions of software are supported if the software is locatable, multiple *configured* versions do not work unless the product supports it.

  If set to true, you can install and manage multiple versions of the same software.

  Applies to swconfig, swinstall, and swverify.

- **`allow_partial_bundles=true`**

  Determines whether to process partial bundles without issuing warnings or notes to log files.

  If true (default), swpackage packages what is available in the source PSF, ignoring missing or ambiguous bundle contents. Bundles are wrappers around products, so you must make sure the bundle contents are placed in the resulting depot, if they are not in the PSF. Otherwise, the missing/ambiguous contents will not affect the system after installation.

  If false, swpackage expects all the bundle contents to be present in the source PSF and to be unique. Every content that is ambiguous or missing gets a NOTE and every bundle that has a missing or ambiguous content gets a WARNING.

  Applies only to swpackage.

- **`allow_split_patches=false`**

  Controls the ability to install or copy part of a patch. Use this option only to resolve critical problems with the assistance of your HP support representative.

When set to the default value of false, installation or copy of a single fileset from a multi-fileset patch automatically includes other, “sibling” filesets that are appropriate, based on the target’s `ancestor` attributes. This behavior applies to any filesets you select directly and to filesets automatically selected to meet dependencies for a patch filesets. Likewise, removing a fileset when this option is false causes sibling filesets to be removed at the same time.

When set to true, this option allows a single patch fileset to be installed, copied, or removed to or from a target without dragging along sibling filesets (that is, filesets that have an ancestor on the target that would usually be loaded if this option was set to false). This allows a target to contain a patch that has been “split” into component filesets. This can create harmful results if one fileset in a sibling group is updated while others remain at an earlier release. Running a swlist on the target after using this option may show more than one active patch active at one time. This makes your system more difficult to maintain and troubleshoot.

Applies to swcopy, swinstall, and swremove.

- **`alternate_source=`**

  Syntax is `host:path`, used when `use_alternate_source` default is set to true.

  By default, this option is not defined. If the host portion is not specified, the local host is used. If the path is not specified, the path sent by the command is used. See the `use_alternate_source` option.

  The protocol sequence and endpoint given by the option `rpc_binding_info_alt_source` are used when the agent attempts to contact an alternate source depot.

  Applies only to swagent.

- **`ask=true`**

  Executes a request script, which asks for a user response.

  The ask option has three possible values;

`true`	(Default for swask) Executes the request script (if one exists for the selected software) and stores the user response in a file named `response`.

false	(Default for swinstall and swconfig.) Does not execute request scripts.
as_needed	Before executing a request script, swask first determines if a response file already exists and executes the request script only if the response file is absent from the control directory.

See “Requesting User Responses (swask)” on page 407 for more information on the swask command and writing request scripts.

Applies to swask, swconfig, and swinstall.

- **auto_kernel_build=true**

  Normally set to true. Specifies whether the removal of a kernel fileset should rebuild the kernel or not. If the kernel rebuild succeeds, the system automatically reboots. If set to false, the system continues to run the current kernel.

  If the `auto_kernel_build` option is set to true, the `autoreboot` option must also be set to true. If the `auto_kernel_build` option is set to false the value of the `autoreboot` option does not matter.

  Applies only to swremove.

- **autoreboot=false**

  Normally set to false, indicating that installation of software requiring a reboot is not allowed from the command line.

  If set to true, this option allows installation or removal of the software and automatically reboots the local host.

  If the `auto_kernel_build` option is set to true, this option must also be set to true.

  Applies to swinstall and swremove.

- **autorecover=false**

  This option permits automatic recovery of original filesets if an installation error occurs. The cost is a temporary increase in disk space and slower performance. The default value of false causes swinstall to overwrite original files as a fileset is updated. If an error occurs during the installation (e.g. network failure), then the original files are lost, and you must reinstall the fileset.

If set to true, all files are saved as backup copies until the current fileset finishes loading. If an error occurs during installation, the fileset's original files are restored, and swinstall continues to the next fileset in the product or the product postinstall script.

When set to true, this option also affects scripts. For example, if a preinstall script fails, this option causes the corresponding unpreinstall script to execute.

Applies only to swinstall.

- **`autorecover_product=false`**

  By default, swinstall overwrites old files. If a load error occurs, the product is marked "corrupt" and you must retry the install.

  When this option is true, swinstall saves all product files as backups until the product finishes loading successfully, then removes the backups. This lets swinstall automatically recover files if the load fails. The trade-off is a temporary increase in disk space use and slower performance

  Note: The autorecover operation does not work properly if any software has pre-install scripts that move or remove files. This includes HP-UX operating system files.

  Applies only to swinstall.

- **`autoremove_job=false`**

  SD-UX stores small amounts of information (such as job status or controller or agent logfiles) about each job. You can display this information from the Job Browser or with swjob. Running very large numbers of jobs may take up significant disk space.

  Setting this option to true prevents SD-UX from storing the job information. The trade-off is that you can no longer display the job information.

  This option is automatically set to true when `run_as_superuser` is set to true.

  Applies to swconfig, swcopy, swinstall, swremove and swverify.

- **`autoselect_dependencies=true`**

  Causes SD-UX to automatically select requisites when software is being selected. At the default value of true, dependent software is automatically selected when you select software with requisites. If set to false, automatic selections are not made to resolve requisites.

  Applies to swconfig, swcopy, swinstall and swverify.

- `autoselect_dependents=false`

  Causes swconfig and swremove to automatically select dependents when software is being selected. When set to true, and any software on which other software depends is selected, SD-UX makes sure that the dependents are also selected. If they are not already selected, they are automatically selected for you. If set to false, dependents are not automatically selected.

  A dependent fileset has established either a prerequisite, corequisite, or exrequisite on the selected fileset. The default value of false prevents automatic selection of dependent software. Specifying true causes SD-UX to automatically select dependent software.

  Applies to swconfig and swremove.

- `autoselect_patches=true`

  Automatically selects the latest patches (based on superseding and ancestor attributes) for a software object that a user selects for a swinstall or swcopy operation. When set to false, the patches corresponding to the selected object are not automatically selected.

  You can use the `patch_filter=` option in conjunction with `autoselect_patches`.

  Applies to swask, swinstall and swcopy.

- `autoselect_reference_bundles=true`

  If true, a bundle that is referenced is installed, copied, or removed along with the software from which the reference is made.

  Applies to swcopy, swinstall and swremove.

- **`check_contents=true`**

  Normally set to true, verify *`mtime`*, *`size`* and *`cksum`* of files.

  If false, the software can be installed without the bundle that contains it.

  Applies only to swverify.

- `check_contents_uncompressed=false`

  When a file is compressed, SD-UX uses it with `check_contents` and `check_contents_use_cksum` to determine whether or not to compute and verify the uncompressed checksum and size of a compressed file. (This option is ignored if the file is not compressed.)

  Since the timestamp of uncompressed contents is meaningless, this option verifies only the timestamp of the file, whether it is compressed or not.

  If set to true and the file is compressed, SD-UX uncompresses the file into memory and computes the checksum and size of the uncompressed contents. Then, the checksum and size of the compressed file and the checksum and size of the uncompressed contents are verified.

  If `check_contents_use_cksum=false`, only the compressed and uncompressed sizes are verified, not the checksums.

  Applies only to swverify.

- **`check_contents_use_cksum=true`**

  Normally true, calculates the checksum of the file being verified, checking the timestamp, size, and checksum.

  If false, turns off the checksum calculation. If `check_contents` is set to true, timestamp and size checking is still performed.

  Applies only to swverify.

- **`check_permissions=true`**

  Normally set to true, verify owner, uid, group, gid and mode attributes of files.

  Applies only to swverify.

- **`check_requisites=true`**

  Normally set to true, verify that the prerequisites and corequisites of filesets are being met.

  Applies only to swverify.

- **`check_scripts=true`**

  Normally set to true, run the vendor-supplied verify scripts when verifying software.

  Applies only to swverify.

- **`check_volatile=false`**

  When set to true, swverify verifies installed files that have the `is_volatile` attribute set. By default, installed volatile files do not have their attributes verified because they are intended to be modified by the customer.

  Applies only to swverify.

- **codeword=**

  Lets you to enter a codeword for the HP-UX licensing procedure. Once entered you need not re-enter the codeword. See “Working with Protected Software” on page 34 for more information.

  Applies to swcopy, swinstall and swlist.

- **`compress_cmd=/usr/contrib/bin/gzip`**

  Specifies the command called by the source agent to compress files before installing, copying, or packaging.

  If you set the `compression_type` option to a value other than `gzip` or `compress`, you must change this path.

  Applies to swagent and swpackage.

- **`compress_files=false`**

  Controls file compression during transfer. When set to false, files are not compressed before transfer from a remote source.

  If set to true, SD-UX compresses files before network transfer if they’re not already compressed. For swinstall, the files are uncompressed after network transfer.

If set to true during swcopy or swpackage, the resulting depots are smaller, unless you also set `uncompress_files` to true.

Applies to swcopy, swinstall and swpackage.

- **`compress_index=false`**

  Enhances performance on slower networks, although it may increase disk space usage due to a larger Installed Products Database and depot catalog. The default of false does not compress INDEX and INFO files. When set to true, INDEX and INFO files are compressed.

  Applies to swinstall, swcopy, swpackage, swmodify, swconfig, and swremove.

- **`compression_type=gzip`**

  Defines the default compression type used by the agent (or set by swpackage) when it compresses files during or after transmission.

  If `uncompress_files` is set to false, the compression type is recorded for each file compressed so that the correct uncompression can later be applied during a swinstall, or a swcopy with `uncompress_files` set to true.

  The `compress_cmd` specified must produce files with the `compression_type` specified.

  The `uncompress_cmd` must be able to process files of the `compression_type` specified unless the format is gzip which is uncompressed by the internal uncompressor (funzip). To use gzip you must load the `SW-DIST.GZIP` fileset (which is optional freeware). If the `SW-DIST.GZIP` fileset is loaded, then you may set the compression options as follows:

  `compress_cmd=/usr/contrib/bin/gzip`

  `uncompress_cmd=/usr/contrib/bin/gunzip`

  `compression_type=gzip`

  Applies to swpackage and swagent.

- **`config_cleanup_cmd=/usr/lbin/sw/config_clean`**

  Defines the script called by the agent to perform release-specific configure cleanup steps.

  Applies only to swagent.

- **`control_files=`**

  When adding or deleting control file objects, this option lists the tags of those control files. There is no supplied default. (Control file objects being added can also be specified in the given product specification file.)

  If there is more than one tag, they must be separated by white space and surrounded by quotes.

  Applies only to swmodify.

- **`controller_source=`**

  Specifies the location of a depot for the controller to access to resolve selections. Setting this option can reduce network traffic between the controller and the target. Use the target selection syntax to specify the location:

  [*host*][:][*path*]

  This option has no effect on which sources the target uses and is ignored when used with an Interactive User Interface.

  Applies to swconfig, swcopy, swinstall, swremove and swverify.

- **`create_target_acls=true`**

  Normally set to true, this default determines whether swpackage creates Access Control Lists (ACLs) in the depot.

  If you set this option to false as superuser, ACLs for each new product being packaged (and for the depot, if it is new) are not created.

  When another user invokes swpackage, it always creates ACLs in the distribution depot. This default has no impact on the ACLs that already exist in the depot. The swpackage command never creates ACLs when software is packaged onto a distribution tape.

  Applies only to swpackage.

- **`create_target_path=true`**

  Normally set to true, creates the target directory if it does not already exist.

  If false, target directory is not created. This option can be used to avoid creating new depots by mistake.

  Applies to swcopy and swinstall.

- **`create_time_filter=0`**

  Controls time settings for cumulative source depots. The default of zero includes all bundles, products, subproducts, and filesets in the source depot as candidates for selection (and autoselection of dependencies and patches), based on the software selections and other options. When set to a time (specified as seconds from epoch), only bundles, products, and filesets (and the subproducts in the product) with a create_time less than or equal to the specified value are available for selection (or autoselection).

  To list the create_time of bundles, products and filesets, use:

  `swlist -a create_time -a create_date`

  Applies to swlist, swcopy, and swinstall.

- **`customer_id=`**

  This number, printed on the Software Certificate, “unlocks” protected software and restricts installation to a specific site or owner. You can enter the number with the `-x customer_id=`option or by using the Interactive User Interface. See the `codeword` option for more information.

  Applies to swinstall, swcopy, swlist.

- **`defer_configure=false`**

  Controls the automatic running of configure scripts after swinstall software selections are installed. The default value of false allows swinstall to automatically run configure scripts. When set to true, swinstall does not run configure scripts. To configure the software later, you must run the swconfig command.

  — Multiple versions of a product will not be automatically configured if another version is already configured. Use the swconfig command to configure multiple versions separately.

— SD-UX ignores this option (treats it as true) when it installs software that causes a system reboot.

— Alternate root directories are not configured.

Applies only to swinstall

- **`distribution_source_directory=/var/spool/sw`**

  Defines the default source depot when the value for the *`source_type`* option is `directory`. You can also use the host:path syntax. The `-s` option overrides this default.

  Applies to swcopy, swinstall, and swpackage.

- **`distribution_target_directory=/var/spool/sw`**

  Defines the default distribution directory of the target depot. The *`target_selection`* operand overrides this default.

  Applies to swacl, swcopy, swlist, swmodify, swpackage, swreg, swremove and swverify.

- **`distribution_target_serial=/dev/rmt/0m`**

  Defines the default location of the target tape device file. The `target_selection` operand overrides this default.

  Applies only to swpackage.

- **`enforce_dependencies=true`**

  When set to true, SD-UX enforces dependencies. swinstall, swcopy, and swconfig do not proceed unless necessary dependencies can be selected, or already exist in the proper state (installed, configured, or available). This prevents SD-UX from installing or copying unusable software. This option also prevents swremove from removing dependent software.

  When set to false, SD-UX checks dependencies but does not enforce them. Corequisite dependencies, if not enforced, may keep the software from working properly. Prerequisite dependencies, if not enforced, may cause the installation or configuration to fail.

  Applies to swconfig, swcopy, swinstall, swremove and swverify.

- **`enforce_dsa=true`**

  When set to the default value of true, SD-UX does not proceed if the disk space required for a software operation is more than the available free space. You can use this option to allow installation into minfree space, or to attempt an install, copy, or package operation even though it may fail because the disk reaches its absolute limit.

  If set to false, space checks are still performed but a warning is issued that the system may not be usable if the disk fills past the minfree threshold. An installation will fail if you run out of disk space.

  Applies to swcopy, swinstall and swpackage.

- **`enforce_kernbld_failure = true`**

  Controls whether or not a failure in either of the kernel build steps (`system_prep` and `mk_kernel`) is fatal to the install session. A failure to build a kernel causes the install process to exit if in non-interactive mode, or to suspend if in an interactive mode.

  If set to false, a failure return from a kernel build process is ignored, and the install session proceeds. The currently running kernel remains in place.

  Applies only to swinstall.

- **`enforce_locatable=true`**

  When set to the default value of true, this option generates an error if a command tries to relocate a non-relocatable fileset. (Relocatable filesets are packaged with the `is_relocatable` attribute set to true.) When set to false, the usual error handling process is overridden, and SD-UX permits the command to relocate the fileset.

  Note that although this option is defined for swverify, there is no swverify behavior associated with the option.

  Applies to swinstall and swverify.

- **`enforce_scripts=true`**

  Controls the handling of errors generated by scripts. If true, and a script returns an error, the command halts, and an error message appears reporting that the execution failed. If false, script-generated errors are treated as warnings, and the command attempts to continue. A warning message appears and reports that the command was successful. Where appropriate, the message identifies the phase in which the error occurred (configure/unconfigure, preinstall/postinstall, preremove/postremove, etc.).

Applies to swask, swconfig, swinstall and swremove.

- **`files=`**

  When adding or deleting file objects, this option can list the path names of those files. There is no supplied default. File objects being added can also be specified in the given product specification file.

  If there is more than one path name, they must be separated by white space and surrounded by double-quotes.

  Applies only to swmodify.

- **`follow_symlinks=false`**

  Do not follow symbolic links that exist in the packaging source; instead, package them as symlinks.

  Applies only to swpackage.

- **`include_file_revisions=false`**

  Normally set to false, controls whether swpackage includes each source file's revision attribute in the product(s) being packaged. Because this operation is very time consuming, the revision attributes are not included by default.

  A value of true for this keyword causes swpackage to execute the what and possibly the ident commands (in that order) to try to determine a file's revision.

  Applies only to swpackage.

- **`install_cleanup_cmd= /usr/lbin/sw/install_clean`**

  The script called by the agent to perform release-specific install cleanup steps immediately after the last postinstall script has been run. For an OS update, this script should at least remove commands that were saved by the `install_setup` script.

  Applies only to swagent.

- **`installed_software_catalog=products`**

  Defines the directory path where the Installed Products Database (IPD, information describing the installed software) is stored. When set to an absolute path, this option defines the location of the IPD.

When this option contains a relative path, the controller appends the path to the path specified by the `admin_directory` option to determine the path to the IPD. For alternate roots, this path is resolved relative to the location of the alternate root. (This option does not affect where software is installed, only the IPD location.)

This option permits the simultaneous installation and removal of multiple software applications by multiple users or multiple processes, with each application or group of applications using a different IPD.

Caution: use a specific `installed_software_catalog` to manage a specific application. SD-UX does not support multiple descriptions of the same application in multiple IPDs.

See also the `admin_directory` option.

Applies to swacl, swask, swconfig, swinstall, swlist, swmodify, swremove, and swverify.

- **`install_setup_cmd=/usr/lbin/sw/install_setup`**

  Defines a script called by the agent to perform release-specific install preparation. For an OS update, this script should copy commands needed for the checkinstall, preinstall, and postinstall scripts to an accessible location while the OS updates the system commands.

  This script is executed before any kernel filesets are loaded.

  Applies only to swagent.

- **`job_title=`**

  When running large numbers of jobs, you may want to add more information to help you identify a specific job. Providing a value for this lets you add an ASCII string that will be displayed along with the ID and other job attributes when you invoke swjob or the job browser.

  Applies to swconfig, swcopy, swinstall, swremove, and swverify.

- **`kernel_build_cmd=/usr/sbin/mk_kernel`**

  This is the script called by the agent for kernel building.

  Applies only to swagent.

- **`kernel_path=/stand/vmunix`**

  The path to the system's bootable kernel. It is passed to the `kernel_build_cmd` via the `SW_KERNEL_PATH` environment variable.

  Applies only to swagent.

- **`layout_version=1.0`**

  Specifies the POSIX layout version to which the SD-UX commands conform when writing distributions and swlist output. Supported values are 1.0 (default) and 0.8. SD-UX for HP-UX version 11.10 and later can read or write either version.

  SD-UX object and attribute syntax conforms to layout version 1.0 of the *IEEE Standard 1387.2, Software Administration (POSIX)*. SD-UX still accepts the keyword names associated with the older layout version, but you should only use layout version 0.8 to create distributions readable by older versions of SD-UX.

  Layout version 1.0 adds significant functionality not recognized by systems supporting only version 0.8, including:

  — Category class objects (formerly the `category` and `category_title` attributes within the bundle or product class).

  — Patch-handling attributes, including `applied_patches`, `is_patch`, and `patch_state`.

  — The fileset `architecture` attribute at the fileset level, which permits you to specify the architecture of the target system on which the software will run.

  In addition to adding new attributes and objects, layout version 1.0 changes the following preexisting 0.8 objects and attributes:

  — Replaces the depot `media_sequence_number` attribute for the media object with a sequence number attribute.

  — Replaces the vendor definition within products and bundles with a `vendor_tag` attribute and a corresponding vendor object defined outside the product or bundle.

  — Pluralizes the `corequisite` and `prerequisite` fileset attributes.

  — Changes the `timestamp` attribute to `mod_time`.

  Applies to swcopy, swlist, swmodify and swpackage.

- **`level=`**

  Specifies a software level for swacl, swlist and swreg.

  For swlist, this option lists all objects down to the specified level. Both the specified levels and the depth of the specified *`software_selections`* control the depth of the swlist output. The supported software levels are:

  — `bundle` -- show all objects down to the bundle level.

  — `product` -- show all objects down to the product level. Also use **`-l bundle -l product`** to show bundles.

  — `subproduct` -- show all objects down to the subproduct level.

  — `fileset` -- show all objects down to the fileset level. Also use **`-l fileset -l subproduct`** to show subproducts.

  — `file` -- show all objects down to the file level (depots, products, filesets, and files).

  — `control_file` -- show all objects down to the control_file level.

  — `category` -- show all categories of available software objects.

  — `patch` -- show all applied patches. (See also the `show_superseded_patches` option.)

  The supported depot and root levels are:

  — `depot` -- show only the depot level (depots that exist at the specified target hosts.

  — `root` -- list all alternate roots.

  — `shroot` -- list all registered shared roots (HP-UX 10.X only).

  — `prroot` -- list all private roots (HP-UX 10.X only).

  For swacl, this option specifies the level of ACLs to view or modify:

  — `host` -- view or modify the ACL protecting the host systems identified by the *`target_selections`*.

  — `depot` -- view or modify the ACL protecting the software depots identified by the *`target_selections`*.

  — `root` -- view or modify the ACL protecting the root file systems identified by the *`target_selections`*.

— `product` -- view or modify the ACL protecting the software product identified by the *`software_selection`*. Applies only to products in depots, not installed products in roots

— `product_template` -- view or modify the template ACL used to initialize the ACLs of future products added to the software depots identified by the *`target_selections`*.

— `global_soc_template` -- view or modify the template ACL used to initialize the ACLs of future software depots or root file systems added to the hosts identified by the *`target_selections`*.

— `global_product_template` -- view or modify the template ACL used to initialize the *`product_template`* ACLs of future software depots added to the hosts identified by the *`target_selections`*.

For swreg, this option defines the level of object to register or unregister.

— `depot` -- depots that exist at the specified target hosts.

— `root` -- all alternate roots.

— `shroot` -- all registered shared roots (HP-UX 10.X only).

— `prroot` -- all registered private roots (HP-UX 10.X only).

Applies to swacl, swlist, and swreg.

- **`log_msgid=0`**

  Adds numeric identification numbers to the beginning of SD-UX log file messages:

  — 0 (default) adds no identifiers to messages.

  — 1 adds identifiers to ERROR messages only.

  — 2 adds identifiers to ERROR and WARNING messages.

  — 3 adds identifiers to ERROR, WARNING, and NOTE messages.

  — 4 adds identifiers to ERROR, WARNING, NOTE, and certain other informational messages.

  Applies to swconfig, swcopy, swinstall, swmodify, swpackage, swreg, swremove, and swverify.

- **`logdetail=false`**

  Controls the amount of detail written to the log file. When set to true, this option adds detailed task information, such as options specified, progress statements, and additional summary information, to the log file.

  Table A-1 shows the possible combinations of `loglevel` and `logdetail` options.

Table A-2 **`loglevel` and `logdetail` Combinations**

Log Level	Log Detail	Information Included
loglevel=0	(not applicable)	No information is written to the log file
loglevel=1	logdetail=false	Only key events are logged. This is the default setting for both options.
loglevel=1	logdetail=true	Event detail as above plus task progress messages. (Setting *`loglevel=1`* is optional because 1 is the default value.)
loglevel=2	logdetail=false	Event and file level messages only. (Setting *`logdetail=false`* is optional because false is the default value.)
loglevel=2 [a]	logdetail=true	All information is logged.

a. This combination duplicates the logfile behavior as HP-UX 10.x releases. Setting both the *`loglevel=2`* and *`logdetail=true`* options is required.

Applies to swconfig, swcopy, swinstall, swreg, swremove and swverify.

- **`logfile=/var/adm/sw/<command>.log`**

  This is the default controller log file for each command. The agent log files are always located relative to the target depot or target root: `/var/spool/sw/swagent.log` and `/var/adm/sw/swagent.log`, respectively.

  Applies to all commands except swacl, swlist and swjob.

- **`loglevel=1`**

  This option controls the log level for events logged to the command log file, the target agent log file and the source agent log file by prepending identification numbers to SD-UX log file messages. This information is in addition to the detail controlled by the `logdetail` option. A value of:

  — 0 -- provides no information to the log files

  — 1 -- enables verbose logging to the log files

  — 2 -- enables very verbose logging to the log files.

  Applies to swconfig, swcopy, swinstall, swmodify, swpackage, swremove and swverify.

- **`match_target=false`**

  If set to true, forces selection of filesets from the source that match filesets already installed on the target system.

  Filesets on the source which specify an installed fileset as an “ancestor” will be selected.

  This option overrides any other software selections.

  Selections cannot be ambiguous.

  Applies only to swinstall.

- **`max_agents=-1`**

  The maximum number of agents that are permitted to run simultaneously. The value of -1 means there is no limit.

  Applies only to swagentd.

- **`max_targets=25`**

  When set to a positive integer, this option limits the number of concurrent install or copy operations to the number specified. As each copy or install operation completes, another target is selected and started until all targets are completed.

  Server and network performance determines the optimal setting; a recommended starting point is 25 (the default value). If you set this option to a value of less than one, SD-UX attempts to install or copy to all targets at once.

  Applies to swcopy and swinstall.

- **`media_capacity=1330`**

  If creating a distribution tape or multiple-directory media such as a CD-ROM, this keyword specifies the capacity of the tape in one million byte units (not Mbytes). This option is required if the media is not a DDS tape or a disk file. Without this option, swpackage sets the size to the default of 1,330 million bytes for tape or to the amount of free space on the disk up to minfree for a disk file.

  SD-UX uses the same format across multiple directory media as it does for multiple serial media, including calculations of the correct size based partitioning of filesets and setting of the `media_sequence_number` attributes.

  Applies only to swpackage.

- **`media_type=directory`**

  Defines the type of distribution to create. The recognized types are `directory` and `tape`. Without this option, swpackage creates a distribution directory (depot) by default.

  Applies only to swpackage.

- **`minimum_job_polling_interval=1`**

  Defines how often, in minutes, the daemon will “wake up” and scan the job queue to determine if any scheduled jobs need to be initiated or if any active jobs need their remote target status cached locally.

  If set to 0, no scheduled jobs will be initiated, and no caching of active jobs will occur.

  Applies only to swagentd.

- **`mount_all_filesystems=true`**

  Normally set to true, the commands automatically try to mount all file systems in the file system table (`/etc/fstab`) at the beginning of the analysis phase and make sure that all those file systems are mounted before proceeding.

  When set to false, no additional file systems are mounted.

  Applies to swconfig, swcopy, swinstall, swremove, swverify.

- **`mount_cmd=/sbin/mount`**

  Specifies the command called by the agent to mount all file systems.

  Applies only to swagent.

- **`objects_to_register=`**

  Defines the default objects to register or unregister. If there is more than one object, they must be separated by spaces.

  There is no supplied default. See also `select_local`.

  Applies only to swreg.

- **`one_liner=<attributes>`**

  Defines the attributes listed in the non-verbose listing.

  If there is more than one attribute, they must be separated by a space and surrounded by quotes.

  `one_liner="revision size title"`

  You must choose which attributes (that is, revision, size, title, etc.) a default listing of software should use. Note: the `tag` attribute is always displayed for bundles, products, subproducts and filesets; the `path` is always displayed for files.

  Any attributes may be chosen but a particular attribute may not exist for all applicable software classes (bundle, product, subproduct, fileset). For example, the software attribute, `title` is available for bundles, products, subproducts and filesets, but the attribute `architecture` is only available for products and filesets.

  In the absence of the `-v` or `-a` option, swlist displays `one_liner` information for each software object (bundles, products, subproducts and filesets).

  Applies only to swlist.

- **`os_name`**

  Specifies fileset selection for an HP-UX update. (This option should always be used with the `os_release` option.) You must specify this option from the command line or when invoking the swinstall GUI.

  This options has the following syntax:

  `os_name`**`=`*****`operating_system`*****`:width`**

  — *`operating_system`* specifies the name of the operating system, such as `HP-UX`. See the *uname*(1) manual page for complete information.

  — *`width`* specifies the word width in bits (either 32 or 64) of the OS to be installed.

  — *`operating_system`* and *`width`* must be separated by a colon (`:`).

  Applies only to swinstall.

- **`os_release`**

  Specifies fileset selection for an HP-UX update. (This option should always be used with the `os_name` option.) You must specify this option from the command line or when invoking the `swinstall` GUI.

  This options has the following syntax:

  `os_release=`***`release`***

  `release` specifies the HP-UX release. Values include:

```
B.10.01
B.10.10
B.10.20
B.10.30
B.11.00
```

  Applies only to swinstall.

- **`package_in_place=no`**

  Setting this option to yes causes swpackage to build the products such that the depot does not actually contain the files that make up a product. Instead, the depot references the original source files used to build a product. This lets you package products in a development or test environment without consuming the extra disk space required to create a distribution depot.

  Applies only to swpackage.

- **`patch_commit=false`**

  Commits a patch by removing files saved for patch rollback. The default value is false. When set to true, this option removes the saved files for the patches specified in the software selections for the command. Once you have run this option on a patch, you cannot remove the patch unless you remove the associated base software that the patch modified.

  Applies only to swmodify.

- **`patch_filter=*.*`**

  Specifies a *`software_specification`* for a patch filter.

  This option can be is used in conjunction with the `autoselect_patches` and `patch_match_target` options to filter the selected patches to meet the criteria specified by *`software_specification`*. The default *`software_specification`* value is `*.*`.

  Note that patch filtering is overridden if you specify software with a command or if you use the `\*` wildcard to select software

  Applies to swask, swcopy and swinstall.

- **`patch_match_target=false`**

  If set to true, this option selects the latest patches (software packaged with the `is_patch` attribute set to true) that correspond to software on the target root or depot.

  The `patch_filter=` option can be used with the `patch_match_target` option.

  Applies to swcopy and swinstall.

- **`patch_one_liner=title patch_state`**

  Use this command to specify the attributes displayed for each object listed when the `-l` patch option is invoked and when no `-a` or `-v` option is specified. The default display attributes are `title` and `patch_state`.

  Applies to swlist and swjob.

- **`patch_save_files=true`**

  Saves patched files, which permits future rollback of patches. When set to false, patches cannot be rolled back (removed) unless the base software modified by the patch is removed at the same time.

  Applies to swinstall.

- **`polling_interval=2`**

  Applies only to interactive sessions. Specifies how often, in seconds, SD-UX polls each target for status information during the analysis and execution phases. When you must operate across wide-area networks, you can increase the polling interval to reduce network overhead. Specifying a high numbers for this option creates longer intervals between polls.

  Applies to swcopy, swinstall and swremove.

- **`preserve_create_time=false`**

  Preserves the original create time when you copy depots, which produces consistent results when you use the copies. The default of false sets the create_time of software bundles, products, and filesets equal to the time swcopy created the new depot. When set to true, the create_time is set to that specified in the source depot from which the current selections were copied. Note that using this option when copying to a master depot can change the objects that are visible when you use the `create_time_filter` option.

  Applies to swcopy.

- **`preview=false`**

  If true, run this command in preview mode only (complete the analysis phase and exit). This option has the same effect as specifying -p on the command line.

  Applies to swcopy, swinstall, swremove, and swconfig.

- **`reboot_cmd=/sbin/reboot`**

  This is the command called by the agent to reboot the system.

  Applies to swagent.

- **`reconfigure=false`**

  This option prevents software that is already in the configured state from being reconfigured. If set to true, configured software can be reconfigured.

  Applies to swconfig.

- **`register_new_depot=true`**

  Normally set to true, a newly created depot is registered on its host. This allows other commands to automatically see this depot.

  If set to false, new depots are not registered. This could allow you to create a private depot on which to test, then later register it with swreg.

  Applies only to swcopy.

- **`register_new_root=true`**

  Causes swinstall to register a newly-created alternate root with the local swagentd, which lets other SD-UX commands see this root.

  If set to false, a new root is not automatically registered. (You can use the swreg command to register the depot later.)

  Applies only to swinstall.

- **`reinstall=false`**

  Prevents SD-UX from re-installing (overwriting) an existing revision of a fileset. If set to true, the fileset are re-installed.

  Applies to swcopy and swinstall.

- **`reinstall_files=false`**

  Controls the overwriting of files, which may enhance performance on slow networks or disks. At the default value of false, SD-UX compares each file in a source fileset to corresponding files on the target system. SD-UX compares the files based on size, timestamp, and (optionally) the checksum. If the files are identical the files on the target system are not overwritten.

  When set to true, SD-UX does not compare files and overwrites any identical files on the target.

  See also the `reinstall` and `reinstall_files_use_cksum` options.

  Applies to swcopy, swinstall and swpackage.

- **`reinstall_files_use_cksum=true`**
  `reinstall_files_use_cksum=false` (swpackage only)

  Controls the use of checksum comparisons when the `reinstall_files` option is set to false. The default value of true causes SD-UX to compute and compare checksums to determine if a new file should overwrite an old file. Use of checksums slows the comparison but is a more robust check for equivalency than size and time stamp.

  If set to false, SD-UX does not compute checksums and compares files only by size and timestamp.

  Applies to swcopy, swinstall and swpackage.

- **`remove_empty_depot=true`**

  When the last product or bundle in a depot is removed, the depot itself is removed.

  If set to false, the depot is not removed when the last product (or bundle) in it is removed. This preserves that depot's ACL.

  Applies only to swremove.

- **`remove_obsolete_filesets=false`**

  This command controls whether swcopy automatically removes obsolete filesets from target products in the target depot. If set to true, swcopy removes obsolete filesets from the target products that were written during the copy process. Removal occurs after the copy is complete. Filesets are defined as obsolete if they were not part of the most recent packaging of the product residing on the source depot.

  Applies only to swcopy.

- **`remove_setup_cmd=/usr/lbin/sw/remove_setup`**

  Defines the script called by the agent to perform release-specific removal preparation. For an OS update, this script invokes the tlink command when a fileset is removed.

  Applies only to swagentd.

- **`retry_rpc=1`**

  This command defines the number of times a lost source connection is retried during file transfers. If set from 1 to 9, the install of each fileset is attempted that number of times. The `reinstall_files` option should also be set to false to avoid installing files that were successfully installed within the fileset.

  This option also applies to the controller contacting the agent. If the agent session fails to start for any reason, the controller tries to recontact that agent for the number of times specified in `retry_rpc`, using the values from the `retry_rpc_interval` option to determine how long to wait between each attempt to recontact the agent.

  Applies to swcopy and swinstall.

- **`retry_rpc_interval={0}`**

  Specifies in minutes the length of the interval for repeated attempts to make a connection to a target after an initial failure. Used in conjunction with the `retry_rpc` option.

  If the number of values in this option equals the value of `retry_rpc`, SD-UX tries to reestablish a source connection for the number of times specified in `retry_rpc`. If the number of values in `retry_rpc_interval` is less than the value in `retry_rpc`, SD-UX repeats the final interval value until the number of retries matches `retry_rpc`.

  For example, if an agent session failed to start and `retry_rpc` was set to 9 and `retry_rpc_interval` was set to {1 2 4 8 15} to allow long waits to handle transient network failures, the controller would attempt to recontact the agent after 1 minute for the first retry, then 2 minutes for the second retry, 4 for the third retry, then 8, then 15 for all additional retries until nine retries were attempted. With these values, a file load failure could cause the operation to pause for 90 minutes (1+2+4+8+15+15+15+15+15). If both options were set to 5, the controller would try to contact the target five times over a 30-minute period.

  Applies to swcopy and swinstall.

- **`reuse_short_job_numbers=true`**

  When assigning job ID numbers, SD-UX uses numbers less than 10,000. Typically, old jobs are removed long before job number 9,999 is reached, so the job number quickly rolls over from 9999 back to 1. When you execute a large numbers of jobs that are not removed before the ID numbers reach 9,999, SD-UX may have performance delays while it searches for unused job numbers.

  Setting `reuse_short_job_numbers` to false causes SD-UX to begin using numbers above 10,000. This avoids possible searching delays and lets the job ID numbers increase to 8 digits (99,999,999) if necessary. (This prevents roll-over from 9999 back to 1, so is usually not desirable.)

  See also the `autoremove_job` option.

  Applies to swconfig, swcopy, swinstall, swremove, and swverify.

- **`rpc_binding_info=ncacn_ip_tcp:[2121] ncadg_ip_udp:[2121]`**

  Determines on what protocol sequence(s) and endpoint(s) the swagentd daemon listens. If the connection fails for one protocol sequence, the next is attempted. SD-UX supports both the tcp and udp protocols on most platforms.

  The value can have the following form:

  — A DCE string binding containing a protocol sequence and an endpoint (a port number). The syntax is:

    `protocol_sequence`**`: [ endpoint ]`**

  — The name of a DCE protocol sequence with no endpoint specified. This syntax is:

    `ncadg_ip_udp` or ***`ncacn_ip_tcp`***

    Since no endpoint is specified, the DCE endpoint mapper *`rpcd`* must be running and is used to find the endpoint registered by swagentd

  — The literal string `all`. This entry means to use all protocol sequences supported by the DCE Remote Procedure Call (RPC) runtime. It should be the only entry in the list. The *`rpcd`* must be running.

  Applies to all commands except swask, swmodify, and swpackage.

- **`rpc_binding_info_alt_source=ncadg_ip_udp:[2121]`**

  Defines the protocol sequence(s) and endpoint(s) used when the agent attempts to contact an alternate source depot specified by the `alternate_source` option. SD-UX supports both the `udp(ncadg_ip_udp:[2121])` and `tcp(ncacn_ip_tcp:[2121])`protocol sequence/endpoint.

  Applies to swagent.

- **`rpc_timeout=5`**

  Relative length of the communications timeout. This is a value in the range from 0 to 9 and is interpreted by the DCE RPC. Higher values mean longer times; you may need a higher value for a slow or busy network. Lower values give faster recognition of attempts to contact hosts that are not up or are not running swagentd.

  Each value is approximately twice as long as the lower one. A value of 5 is about 30 seconds for the `ncadg_ip_udp` protocol sequence. This option may be ignored when using the `ncacn_ip_tcp` protocol sequence.

  Applies to all commands except swmodify and swpackage.

- **`run_as_superuser=true`**

  This option controls SD-UX's nonprivileged mode. This option is ignored (treated as true) when the invoking user is super-user.

  At the default value of true, SD-UX operations are performed normally, with permissions for operations either granted to a local super-user or set by ACLs. (See Chapter 9, "SD-UX Security," on page 255 for details on ACLs.)

  When set to false and the invoking user is local and is not super-user, nonprivileged mode is invoked:

  — Permissions for operations are based on the user's file system permissions.

  — ACLs are ignored.

  — Files created by SD-UX have the uid and gid of the invoking user, and the mode of created files is set according to the invoking user's umask.

Nonprivileged mode is intended only for managing applications that are specially designed and packaged. This mode cannot be used to manage the HP-UX operating system or patches to it. This option is not compatible with remote operations. Setting this option to true forces the `autoremove_job` option to true. For a full explanation of nonprivileged SD-UX, see Chapter 12, “Nonprivileged SD,” on page 413.

See also the `admin_directory` option.

Applies to all commands except swagent, swagentd, and install-sd.

- **`select_local=true`**

  Normally set to true, selects the default depot or installation directory of the local host as the target of the command.

  Applies to swacl, swconfig, swcopy, swinstall, swlist, swreg, swremove, swverify.

- **`show_superseded_patches=false`**

  If false, swlist will not display superseded patches. To see them, you must set this option to true. Even if you explicitly swlist the superseded patch, it will not display unless this option is true.

  Applies only to swlist.

- **`software=`**

  Defines the default *`software_selections`*.

  There is no supplied default. If there is more than one *`software_selection`*, they must be surrounded by brackets { } or quotes. Software is usually specified in a software_selections input file, as options on the command line or in the GUI or TUI.

  Applies to all commands except swreg.

- **`software_view=products`**

  Indicates which software view is to be used in the GUI. It can be set to `products`, `all_bundles`, or a `bundle category` tag (shows only bundles of that category). The default view is all_bundles plus products that are not part of a bundle.

  Applies to swcopy, swinstall, swlist and swremove.

- **`source=`**

  Specify a source to automatically bypass the GUI and CLI source selection dialog box. This has the same effect as the `-s` *source* command line option. Specify the source using the following syntax:

`[`*`path`*`]`

  Applies to swcopy and swinstall.

- **`source_cdrom=/SD_CDROM`**

  Defines the default location of the source CD-ROM. The syntax is:

`[`*`host`*`][:][`*`path`*`]`

  Applies only to swinstall.

- **`source_depot_audit=true`**

  If both source and target machine are updated to SD-UX revision B.11.00 or later, the system administrator at the source depot machine can set this option to track *`which`* user pulls *`which`* software from a depot on the source machine and *`when`* the software is pulled.

  A user running swinstall or swcopy from a target machine cannot set this option; only the administrator of the source depot machine can set it.

  When `source_depot_audit` is set to its default value of true, a `swaudit.log` file is created on the source depot (for writable directory depots) or in `/var/tmp` (for tar images, CD-ROMs, or other nonwritable depots).

  To view, print, or save the audit information, invoke the `swlist` interactive user interface by typing:

  ```
  swlist -i -d
  ```

  You can view audit information based on language preference, as long as the system has the corresponding SD-UX message catalog files on it. For example, you can view the source audit information in Japanese during one invocation of swlist, then view the same information in English at the next invocation.

  Applies to swagent.

- **`source_file=psf`**

  This keyword defines the default *`product_specification_file`* to read as input to the packaging or swmodify session. It may be a relative or absolute path.

  Applies to swpackage and swmodify.

- **`source_tape=/dev/rmt/0m`**

  Defines the default tape location, usually the character-special file of a local tape device. You can also use the **`host:path`** syntax, but the host must match the local host. The `-s` option overrides this value.

  Applies to swcopy and swinstall.

- **`source_type=directory`**

  The default source type (choices are `cdrom`, `file`, `directory`, or `tape`) that points to one of the next three options. The source type derived from the `-s` *`source file`* option overrides this default.

  The `cdrom` and `tape` values apply to swcopy and swinstall. The `file` value applies only to swpackage.

- **`system_file_path=/stand/system`**

  The path to the kernel's template file. The path is passed to the `system_prep_command` via the `SW_SYSTEM_FILE_PATH` environment variable.

  Applies only to swagent.

- **`system_prep_cmd=/usr/lbin/sysadm/system_prep`**

  The kernel build preparation script called by the agent. This script must do any necessary preparation so that control scripts can correctly configure the kernel that is about to be built.

  Applies only to swagent.

- **`targets=`**

  There is no supplied default (see also `select_local`). If there is more than one target, they must be separated by spaces. Targets are usually specified in a target input file, as options on the command line or in the GUI.

  Applies to all commands.

- **`target_type`**

  See `media_type`.

- **`uncompress_cmd=`**

  This is the command called by the source agent to uncompress files when installing, copying or packaging.

  This command processes files that were stored on the media in compressed format. If the `compression_type` of the file is `gzip`, the internal compression (`funzip`) is used instead of the external uncompress command.

  Applies to swagent and swpackage.

- **`uncompress_files=false`**

  When set to true, files are uncompressed using the current `uncompress_cmd` before storing them on the target depot.

  Only one of the `uncompress_files` and `compress_files` options may be set to true during a swpackage session.

  The `uncompress_files` option may not be set to true if `package_in_place` is set to true or if the `media_type` is set to `tape`.

  Applies to swcopy and swpackage.

- **`use_alternate_source=false`**

  At the default value of false, swinstall or swcopy begins an analysis or task with a request that includes information describing the source binding and depot path for the local host to use as the software source.

  If true, the local host uses its own configured value. On the local host, the agent's configured value for `alternate_source` is specified in **`host:/path`** format. If this value contains only a path component (for example, **`alternate_source=:/path`**), the agent applies this path to the file system of its own local host.

  If only the host component exists (for example, **`alternate_source=host`**), the agent applies the controller-supplied path to this host. If there is no configured value at all for the `alternate_source`, the agent applies the controller-supplied path to its own local host.

  Applies to swcopy and swinstall.

- **`verbose=`**

  By default, the command sends output to `stdout` for task summary messages. Alternatively, the verbose option can be set to 0 for session level messages (no output to `stdout`) or (for swpackage and swmodify) to 2 for file level messages.

  Error and warning messages are always written to `stderr`.

  For the swlist command, a verbose listing includes all attributes that have been defined for the appropriate level of each *`software_selection`* operand. The attributes are listed one per line, prefaced by the attribute keyword.

  The `-v` option overrides this default, if it is set to 0.

  Applies to all commands.

- **`write_remote_files=false`**

  Prevents file operations on remote (NFS) file systems. All files destined for installation, copy, removal, or packaging on targets on a remote (NFS) file systems are skipped.

  If set to true and if the superuser has write permission on the remote file system, the remote files are not skipped.

  Applies to swconfig, swcopy, swinstall, swpackage, and swremove.

B Troubleshooting

This appendix explains how SD-UX error messages are used, reviews the SD-UX error logging process. lists common problems you might encounter, and suggests how to resolve them.

Table B-1 **Chapter Topics**

Topics:
"Error Logging" on page 462
"Common Problems" on page 464

Error Logging

All SD-UX commands (except swlist and swacl) log error messages, summary information about the session, and operation details to a command-specific logfile located (by default) in `/var/adm/sw/<`*`command`*`>.log`. For example, if you wanted to examine the logfile for swinstall, you would look in the file `/var/adm/sw/swinstall.log`. You can also examine target agent logfiles for a current session from the swinstall, swcopy, or swremove GUIs.

If you have log-in access to a target host, you can see its *agent* logfile(s) directly. The location of the agent logfile varies, depending on the type of target:

- `/var/adm/sw/swagent.log` when operating on a host’s primary root.
- `/<root_path>/var/adm/sw/swagent.log` for an alternate root.
- `/<depot_path>/swagent.log` for a target or source depot.

The default location of a host’s daemon logfile is `/var/adm/sw/swagentd.log`. This logfile contains information for problems starting agents, particularly for problems where you have access denied to a depot or root.

NOTE When both the source and target machine are updated to HP-UX 10.30 or later, the system administrator at the source depot machine can track which user pulls which software from a depot and when the software is pulled. Refer to the `source_depot_audit` option in Appendix A, “Command Options,” on page 421 or “Source Depot Auditing” on page 160.

Error Messages

SD-UX error messages indicate that a problem occurred that will influence the overall outcome of an operation.

For example, if a target in an install session fails the analysis phase due to insufficient disk space, you would find the following error message in the agent log file:

```
ERROR:The estimated disk space used on filesystem "/" is
14104 Kbyte blocks. This operation will exceed the
minimum free space for this disk. You should free up at
least 2280 Kbyte blocks to avoid installing beyond this
threshold of available user disk space. If you are
running interactive "swinstall", you must return to the
Selection Window and Unmark this target before using
"swremove" to free disk space.
```

Warning Messages

Warning messages let you know that something unexpected and potentially undesirable occurred. A warning does not prevent the SD session from continuing. Warning messages during analysis of an interactive session give you the chance to continue or stop.

For example, if the fileset SD-DATABASE.SD-DATABASE2 is being installed in multiple locations on a target system, you would find the following warning message in the agent log file:

```
WARNING:A version of fileset
"SD-DATABASE.SD-DATABASE2,r=9.00.1C"
is already installed in another location (see previous
lines). Installing this version will create multiple
installed versions. This new multiple version will be
installed because the "allow_multiple_versions" option is
set to "true".
```

Notes

Notes are used to notify you of an event that is not erroneous, unexpected or undesirable, but that you should be aware of:

```
NOTE:The fileset "SD-DATABASE.SD-DATABASE1,r=9.00.1C" is
already installed. If you wish to reinstall this
fileset, change the "reinstall" option to "true".
```

Common Problems

This section presents a selection of problems you might encounter and how to resolve them:

Table B-2 Common Problems

Problem
Cannot contact target host’s daemon or agent
GUI won’t start or missing support files
Access to an object is denied
Slow network performance
Connection timeouts and other WAN problems
Disk space analysis is incorrect
The packager fails
Daemon logfile is too long
Cannot read a tape depot
Installation fails
swinstall or swremove fails with a lock error

Cannot Contact Target Host’s Daemon or Agent

If you see the following error message:

```
ERROR: Could not contact host <hostname>. Make sure the
hostname is correct.
```

it means that the hostname you specified could not be found in the hosts database. Make sure you have typed the hostname correctly (you can use the nslookup command to verify hostnames). If the target hostname is not in the hosts database, but you know its network address, you can use it (in standard “dot” notation) in place of the hostname.

If you see this error message:

```
ERROR:Remote Procedure Call to a daemon has failed.
Could not start a management session for <target>.
Make sure the host is accessible from the network,
and that its daemon, swagentd, is running. If the
daemon is running see the daemon logfile
on this target for more information.
```

it means SD-UX could not contact the daemon program on a specific target system. Note that this may occur even if you haven’t specified any targets, for example, if the daemon on your local host is not running.

Resolution

If the SD-UX daemon/agent is not installed on a given target system, you must install it before you can use SD-UX.

If you’ve verified that the daemon/agent component has been installed on a target system and you still have trouble contacting it, check to see that the daemon is running:

1. On the target system, type:

   ```
   ps -e | grep swagentd
   ```

2. If the daemon does not appear to be running, you can start it by typing (as root on the target system):

   ```
   /usr/sbin/swagentd
   ```

3. If you attempt to start a daemon when one is already running, you will see a message about the other daemon; this is harmless.

   You can also kill and restart a currently running daemon by typing:

   ```
   /usr/sbin/swagentd -r
   ```

Other possible causes for this problem are listed in the section "Connection Timeouts and Other WAN Problems" on page 472.

TIP An easy way to determine if a target system has the SD-UX daemon installed and running is to type:

```
/usr/sbin/swlist -l depot @ <one or more target hostnames>
```

which will attempt to contact each target to get a list of registered depots. Those targets which have the SD-UX daemon installed will report either:

```
# Initializing...
# Target <hostname> has the following depot(s):
# <...insert list of depots...>
```

or

```
# Initializing...
WARNING: No depot was found for <hostname>.
```

For more information on daemon activity, see the daemon logfile in `/var/adm/sw/swagentd.log`.

GUI Won't Start or Missing Support Files

You can start the GUI in these ways:

- For swinstall, swcopy, or swremove, type the command with no additional options or arguments.
- Include the `-i` option with any other options and arguments when you type the command on the command line. (Required for swlist.)
- For the Job Browser, type **sd** on the command line.

When using the GUI, you might encounter these problems:

- Can't open the display or display is set incorrectly
- Missing GUI support files

Resolution

If you have invoked the GUI on a remote system, you may see the following error messages:

```
Xlib: connection to <display> refused by server
Xlib: Client is not authorized to connect to Server
Error: Can't Open display.
```

Check that you have set the `$DISPLAY` environment variable correctly on the remote system to identify your display. If it is correct, you may have to enable the remote host to make connections to your X server via the `xhost(1)` command or by modifying your `/etc/X*.hosts` file.

If you see the error message:

```
swinstall: Error: cannot read file:
          /usr/lib/sw/ui/smc_install_copy.ui
```

— *or* —

```
swremove: Error: cannot read file:
/usr/lib/sw/ui/smc_remove.ui
```

the system is telling you that the file `/usr/lib/sw/ui/smc_install_copy.ui` must be installed on the system to run either swinstall or swcopy interactively or that the `/usr/lib/sw/ui/smc_remove.ui` file must be installed to run swremove. Make sure that the directory `/usr/lib/sw/ui` exists and includes the requested file. If the file does not exist, you must reinstall the `SD-CMDS` fileset from your OS media.

Access To An Object Is Denied

Denial of access to SD-UX objects may have a number of causes, including:

- ACL permissions
- Inter-host secrets
- Working with image copies of depots

Resolution

Generally, when SD-UX denies access to an object, a message tells you that you do not have the required access permission. Yet, it may be unclear which object is not accessible. For example, when you use swcopy to copy a product from system A to a depot, SD-UX checks these ACLs:

1. If the destination depot does NOT exist, the host ACL is checked to verify that the user has "insert" permission.
2. If the destination depot does exist, the depot ACL is checked to verify that the user has write permission.
3. The source depot's ACL is checked to make sure the user has read permission on the source depot.
4. The source product's ACL is also checked to make sure that the user and the destination system both have read access to the product.

If any of these access permissions is absent, the whole operation is disallowed, and you must read the error message carefully to understanding the exact cause. To see more about what type of security or access problems exist, see the daemon log file on the target system: `/var/adm/sw/swagentd.log`

The Effects of ACL Modifications

The default ACLs make it fairly easy to administer ACLs, but do not always give the desired level of access control. When you change an ACL to restrict access, especially by removing the *`any_other`* read permission, you may restrict access in unexpected ways. Host entries are required for any destination systems for swcopy and swinstall operations.

See Chapter 9, "SD-UX Security," on page 255 for a full discussion of the access tests performed or each operation.

Do Not Modify ACL Files Without swacl

Since SD-UX stores ACLs in the file system as plain text files, you may try to edit them with a conventional editor. This can lead to unexpected corruption of the ACL. Most cases of this corruption simply result in a message indicating the corruption, but inserting additions to the ACL file without updating the *num_entries* value can result in unreported problems and cause SD-UX to deny access. A common failure could occur, for instance, if a you inserted user entry in the ACL file. This could push the *any_other* entry down beyond the *num_entries* limit. The ACL manager would never read the *any_other* entry, and you would have access problems. The best guard against this situation is to always use the swacl command to manipulate ACLs.

Inter-host Secrets

The default `/var/adm/sw/security/secrets` file contains a single entry:

```
default      -sdu-
```

If you wish to explicitly name all hosts from which controllers can be run, you must replace the *-sdu-* with a different default secret, or eliminate the entire entry. See Chapter 9, “SD-UX Security,” on page 255 for a thorough discussion of the secrets file.

The controller (for swinstall, swcopy, etc.) looks up the secret for the system on which it runs and passes it in an encrypted form to its agent. The agent receiving a request from the controller looks up the secret for the host from which the call comes, encrypts it, and compares the encryption to that provided by the controller. If the two secrets do not match, access is denied. If you have problems with this mechanism, make sure that all systems have matching entries. You can also revert to the old secrets file (`/etc/newconfig/sd/secrets` on 9.x and `/usr/newconfig/var/adm/sw/security/secrets` on 10.x) on all hosts, or simply copy a single secrets file to all hosts.

Working With Depot Images

You may encounter a problem in using cp, tar, cpio, dd, and other commands to copy images of depots for use on other systems. Depot and product ACLs in the image have built-in knowledge of the host on which the depot originated. In particular, an ACL default *realm* will be wrong and local users will be confused with users on the originating host. For example, attempts to add local users to the access list will, in fact, grant access to remote users. *There is no way to alter the default realm of an ACL from that set when it is created.*

Another common problem with such images occurs if you import them to systems that cannot resolve all the hostnames (see *resolver*(4) and *nslookup*(1)) that exist in the ACLs.

If your purpose is to create a “staged” installation, use swcopy to propagate the depot. This creates new ACLs, based on local templates, for each instance of the depot.

If the sole intent of a depot is for such image distribution, you may wish to set the swpackage `create_target_acls` option to false to prevent ACL creation on the depot and products during the swpackage operation. This option creates tape and CD-ROM images. Depots and products without ACLs grant the local superuser all privileges, while all other users and systems have read access. Note that when you copy or install this ACL-less depot with swcopy or swinstall, the copies (installations) are automatically protected by ACLs based on templates on the destination host.

Slow Network Performance

When using swinstall or swcopy in an environment where network bandwidth is the “bottleneck,” the file transfer rate between source and target can become very slow.

Resolution

The `compress_files=true` option compresses files transferred from a source depot to a target. This can reduce network usage by approximately 50%; the exact amount of compression depends on the type of files. Binary files compress less than 50%, text files more.

The greatest throughput improvements are seen when transfers are across a slow network (approximately 50kbyte/sec or less), and the source depot server is serving a few target hosts at a time.

NOTE This option should be set to true only when network bandwidth is clearly restricting total throughput. If this option is used with a fast network or with a depot server simultaneously connected to many target hosts, this option can actually reduce overall throughput or performance, unless the source depot is already compressed.

If it is not clear that this option will help in your situation, compare the throughput of a few install or copy tasks (both with and without compression) before changing this option value.

See Chapter 8, “Reliability and Performance,” on page 237 for more information about performance options.

Connection Timeouts and Other WAN Problems

Low-throughput, wide-area networks can cause SD-UX to encounter time-out problems when establishing and maintaining network connections with remote agents on other systems.

If you see the following messages:

```
ERROR:A Remote Procedure Call to a daemon has failed.
Could not start a management session for <target>.
Make sure the host is accessible from the network, and
that its daemon, swagentd, is running.  If the daemon is
running see the daemon logfile on this target for more infor
mation.
```

or

```
ERROR: Could not perform the requested operation for
<target>, possibly due to a network communications
failure. Check that the host is still accessible from
the network.
```

and you have verified that the system is up and the daemon program (swagentd) is running on it, it may be that network delays are causing the connection to time-out.

Resolution

Increase the time-out value used by SD-UX when performing Remote Procedure Calls (RPCs) by specifying a higher value for the `rpc_timeout` option, either via the command line or in the defaults file. RPC time-out values range from 0 to 9, with 9 being the longest time-out. The default RPC time-out value is 5. Note that these values do not represent any specific time units. See Appendix A, “Command Options,” on page 421 for more information on the `rpc_timeout` option.

Increasing the `rpc_timeout` can also help in situations where the target agents in an install or copy session are timing out when trying to contact the source agent. This problem is indicated by the following error messages in the agent log file:

```
ERROR: Could not open remote depot/root <path> due to
an RPC or network I/O error.
ERROR: Cannot open source.  Check above for errors, as
well as the daemon logfile on the source host (default
location:/var/adm/sw/swagentd.log).
ERROR: Cannot continue the Analysis Phase until the
previous errors are corrected.
```

Another factor that can affect RPC timeouts on a slow network is the choice of network protocol. SD-UX supports both UDP- and TCP-based communication (the default is TCP). TCP communication is more reliable on a WAN because it is connection-based. SD will fall back to a UDP connection if the TCP connection fails for some reason. The default binding can be set with the `-x rpc_binding_info` option.

Note that the daemon program (swagentd) listens for both UDP- and TCP-based RPCs by default. See Appendix A, “Command Options,” on page 421 for more information on the `rpc_binding_info` option.

A final WAN-related issue may arise when using the interactive GUI. During the analysis and execution phases of an interactive session, each target agent is periodically polled for up-to-date status information. The `polling_interval` option can be used to control the number of seconds that elapse between successive status polls of a given target system. On networks where even this minor data transfer is a problem, you can increase this polling interval, thus decreasing the frequency of polling, and reducing an interactive session’s overall demands on the network. See Appendix A, “Command Options,” on page 421 for more information on the `polling_interval` option.

Disk Space Analysis Is Incorrect

Your installation or copy operation runs out of space even though the disk space analysis succeeded. Upon further checking, you find that the results of the disk space analysis differ from the actual space available.

Resolution

Possible causes of this problem:

- A control script associated with the installation has consumed disk space by creating or copying additional files that aren't accounted for during analysis.
- Your target systems were not idle when the analysis was done and some other activity (unrelated to SD-UX) was consuming disk space.
- The depot from which the product was installed or copied was created by swpackage with the `package_in_place` option set to true, and source files have been modified since the product was packaged. The swverify command can be used to diagnose this problem.

Packager Fails

A swpackage operation may fail because of the incorrect use of the `end` keyword in the Product Specification File (PSF).

Resolution

The `end` keyword marks the end of a depot, vendor, product, subproduct or fileset specification in a PSF. It requires no value and is optional. However, if you use it and it is incorrectly placed, the specification will fail. Check to make sure, if you use it, there is an `end` keyword for every object specification (especially the last one).

Command Logfile Grows Too Large

If you want to reduce the contents of a SD-UX command logfile, follow this procedure:

Resolution

To reduce messages to a minimum, set the `verbose` command option to 0 in one of the option files or by using the `-x` option on the command line. For example, entering `-x spackage.verbose=0` on the command line when you run swpackage would reduce the number of entries to the swpackage log to a minimum. See Appendix A, “Command Options,” on page 421 for details about setting options.

Daemon Logfile Is Too Long

If you want to shorten (truncate) the SD-UX daemon logfile because it is getting too long, follow this procedure:

Resolution

If the daemon is currently running, DO NOT remove its logfile. The running daemon continues to log messages to its logfile even after you’ve removed it, causing any subsequent information to be lost. Also, the disk space used by the logfile will not be freed as long as the daemon is running.

Instead, truncate the logfile by typing (as root):

```
echo > /var/adm/sw/swagentd.log
```

This replaces the previous data in the log with an empty string.

If you inadvertently remove the daemon logfile while it is running, you must kill and restart the daemon if you want to see subsequent daemon log messages and free up the disk space used by the logfile. You can stop (kill) a daemon by typing:

```
usr/sbin/swagentd -k
```

You can also kill and restart a currently running daemon by typing:

```
usr/sbin/swagentd -r
```

Cannot Read a Tape Depot

If you are trying to access a tape depot and see the following error message in the daemon logfile, it means that the tape is either corrupt or is not in SD-UX format.

```
ERROR:The INDEX file on the source did not exist or could
not be read.
ERROR:The target <depot_path> could not be opened.
```

Resolution

Make sure that you have correctly specified the tape device and that the correct tape is in the drive. SD-UX only reads tapes that are in SD-UX format. For example, SD-UX does not read update format tapes.

Installation Fails

An installation may fail while only part way through the process.

Resolution

SD-UX gives you several restart options:

- Re-execute the same command from the command line.
- Recall the session file `swinstall.last` that was automatically saved for you. (See “Session Files” on page 61.)
- Reset the checkpointing options.

  By default, SD-UX checkpoints to the *fileset* level, meaning that the operation will start transferring files with the last fileset to be attempted. By setting the `reinstall_files` option to false, SD-UX restarts distribution and installation with the *file* that was last attempted. (SD-UX does not support checkpointing below the file level.)

  You can override all checkpointing by setting both the `reinstall` and `reinstall_files` options to true. See Appendix A, “Command Options,” on page 421 for more information.

Swinstall or Swremove Fails With a Lock Error

Swinstall or swremove fails with the following message:

```
Cannot lock “/” because another command holds a conflicting
lock. The process id of that command is ####.
```

Resolution

Another SD command is running that prevents the swinstall or swremove command from running. Wait for that command to finish and try again.

C Replacing or Updating SD-UX

This appendix describes how to replace or update SD-UX using the install-sd command.

Table C-1 Chapter Topics

Topics:
"Re-installing SD-UX" on page 480
"Replacing an Unusable Version of SD-UX" on page 482
"Installing a Newer Version of SD-UX" on page 483

Re-installing SD-UX

The software product called `SW-DIST` provides all SD-UX functionality, commands, and tools. This product is included on your HP-UX 11i media.

If the files that make up `SW-DIST` are deleted or corrupted, you may need to re-install the product. The install-sd command lets you install the SD-UX product from HP-UX 11i media or a depot. This command also installs any SD-UX patches that exist in the source depot.

- The `install-sd` command is not supported on HP-UX versions 10.20 or 11.00.
- You need the 11i version of `SW-DIST` to install or copy any HP-UX software that has been packaged in the 11i SD-UX format.
- The update-ux command replaces the swgettools script used by previous versions of SD-UX for OS updates.

For More Information

For complete instructions for updating HP-UX, see:

- *HP-UX 11i Installation and Update Guide*
- *update-ux* (1M) manpage

These documents are available on your HP-UX Instant Information CD-ROM and in the HP-UX 11i section of:

`http://docs.hp.com`

Prerequisites

The install-sd command and an accompanying `swagent.Z` file require at least 2 MB of free space in the `/var/tmp` directory. If there is not enough space in this directory, install-sd will fail. To determine if `/var/tmp` has adequate space, enter:

```
bdf /var/tmp
```

Using install-sd

Syntax

`install-sd -s` *`source_depot_location`*

Options and Operands

The *`source_depot_location`* option specifies an absolute path to the source media location. Possible media locations are:

- A local directory
- A CD-ROM mount point that has an SD-UX media CD-ROM loaded
- A remote system (or host) and depot combination, which you must specify with this syntax:

  `system_name``:`*`/depot_path`*

  For example:

  `swtest:/var/spool/sw`

Command Notes

- The command returns a value of 0 to indicate successful completion and a value of 1 to indicate an error.
- An install-sd session writes messages for major tasks and the begin and end of each session. All WARNING and ERROR conditions are written to stderr.
- Detailed events are logged to `/var/adm/sw/install-sd.log`

Example

`install-sd -s swtest:/var/spool/sw`

Replacing an Unusable Version of SD-UX

If the version of SD-UX on the target system is unusable, you must first load install-sd and the `swagent.Z` file onto your system into `/var/tmp`, then use install-sd to re-install `SW-DIST`. The install-sd utility ships in the `catalog/SW-DIST/pfiles` directory.

Use `cp` (if you are copying from a local CD-ROM) or `rcp` (if you are copying from a software depot on a remote system) to load install-sd onto your system.

For example, to load install-sd from a local CD-ROM mounted at `/SD_CDROM` into `/var/tmp`:

Step 1. Copy install-sd onto your system from the CD-ROM:

```
cp /SD_CDROM/catalog/SW-DIST/pfiles/ \
   install-sd /var/tmp
```

Step 2. Copy the `swagent.Z` file from the CD-ROM:

```
cp /SD_CDROM/catalog/SW-DIST/pfiles/ \
   swagent.Z /var/tmp
```

Step 3. Make install-sd executable:

```
chmod +x /var/tmp/install-sd
```

Step 4. Execute install-sd:

```
/var/tmp/install-sd -s /SD_CDROM
```

The SW-DIST product then installs itself onto your system from the

CD-ROM.

Installing a Newer Version of SD-UX

If you want to install a newer version of SD-UX on your system and`/usr/sbin/install-sd` is not yet on your system, use this procedure.

(In both steps, *`source_depot_location`* is the absolute path to the depot or media that contains the newer version of SD-UX.)

Step 1. As root, enter:

```
/usr/sbin/swinstall -r -s \
source_depot_location\SW-DIST.SD-UPDATE \
@ /var/adm/sw/install-sd.root 2>/dev/null
```

Step 2. Install the newer version:

```
/usr/sbin/install-sd -s source_depot_location
```

NOTE This `install-sd` command will not be available in future releases.

D Software Distributor Files and File System Structure

This chapter contains information on key Software Distributor files.

For additional information, refer to the following manual reference pages:

sd(5)	For most current information on Software Distributor files
sd(4)	For file layouts of all Software Distributor files.
swpackage(4)	For file layouts of Software Distributor files created during packaging.

Table D-1 Chapter Topics

Topics:
"Agent File System Structure" on page 486
"Software Distributor Controller File System Structure" on page 489
"Installed Products Database" on page 490

Agent File System Structure

The agent component is organized as follows:

Table D-2 Agent Component

`/dev/rmt/Om`	Default location of the target tape device file
`/usr/contrib/bin`	Location of the gzip executables used in file compression
`/usr/lbin/swagent`	The SD-UX agent
`/usr/lbin/sw`	Directory containing utilities used by swinstall and swremove
`/usr/lbin/sw/control_utils`	File containing common utilities used by SD control scripts.
`/usr/sbin`	Directory that contains the Software Distributor daemon (and all other executables)
`/usr/lib/sw/examples`	Directory that contains various example packages and PSF files
`/usr/lib/sw/sys.defaults`	File that lists all options and their default values
`/var/adm/sw`	Directory that contains all the data for the Software Distributor product and the default location of logfiles
`/var/spool/sw`	Default directory of the local Software Distributor depot
`/var/adm/sw/defaults`	Software Distributor system-wide defaults file
`/var/adm/sw/host_object`	List of depots registered at the local host

Table D-2 Agent Component (Continued)

`/var/adm/sw/host_object_np`	List of depots registered at the local host during nonprivileged mode
`/var/adm/sw/products`	The Installed Products Database (IPD), a series of files and subdirectories that contain information about all products installed under the root (`/`) directory
`/var/adm/sw/queue`	Directory that contains the Jobs database.
`/var/adm/sw/save`	Directory that is SD's save area for patches
`/var/adm/sw/save_custom`	Directory that is a custom save area for patches
`/var/adm/sw/security`	Directory that contains the host Access Control List (ACL), all default ACLs, and the secrets file
/var/adm/sw/swagent.log	Agent logfile containing details on installed software operations
`/var/adm/sw/swagentd.log`	Daemon log file containing details on host and security operations
`/var/adm/sw/sw<task>.log`	Controller logfile containing a summary of each job, where `<task>` is one of these values: • `install` • `remove` • `config` • `modify` • `package` • `reg` • verify

Table D-2 Agent Component (Continued)

`/var/adm/sw/tmp`	Directory for temporary files
`/var/home/USER_NAME`	Default location for admin_directory during nonprivileged mode
`$HOME/.swdefaults`	File containing user-specified default values. If this file does not exist, Software Distributor looks for user-specific defaults in `$HOME/.sw/defaults`

Software Distributor Controller File System Structure

The controller file system structure is comprised of all files in agent (see "Agent File System Structure" on page 486) plus the following files:

Table D-3 **Controller File System Structure**

`/usr/lib/sw/help`	Directory that contains the help files for on-line help
`/usr/lib/sw/ui`	Directory that contains the description files used by the GUIs
`/usr/lib/X11/app-defaults`	X11 resource definitions for the GUIs
`/usr/lib/nls/msg/$LANG/sw*.cat`	Message catalogs for the daemon, agent, and shared messages
`/usr/newconfig/var/adm/sw`	Data files that are conditionally copied into `/var/adm/sw`.
`/var/adm/sw/queue`	Directory that contains all the data for jobs
`/var/adm/sw/sw<task>.log`	Controller logfile
`/var/adm/sw/defaults.hosts`	System-level *defaults. hosts* file for the GUIs
`/var/adm/sw/.sdkey`	Key file that enables the remote operations GUI
`/var/adm/sw/target_hosts`	Location of the cache file for each target host
`/var/adm/sw/ui/preferences`	Directory that stores the GUI user view preferences

Installed Products Database

Software Distributor commands keep track of installations, products, and filesets on the system with the Installed Product Database (IPD). Located in the directory `/var/adm/sw/products`, the IPD is a series of files and subdirectories that contain information about all the products that are installed under the root directory (`/`). This information includes all the attributes describing the products, filesets, and files. The swinstall, swconfig, and swremove tasks automatically add to, change, and delete this IPD information as the commands are executed.

You cannot manually edit the IPD files, but swmodify lets you change local IPD and local depot catalog information.

The equivalent IPD files for a depot are called catalog files. When a depot is created or modified using swcopy, catalog files are built (by default in `/var/spool/sw/catalog`) that describe the depot and its contents.

The IPD also contains a `swlock` file that manages simultaneous read and/or write access to software objects, and ACLs.

For More Information

- “Modifying the IPD (swmodify)” on page 115

Glossary

NOTE: A glossary term appears in boldface when defined for the first time in the text of this manual. *Italicized* terms in the following glossary refer to other terms in the glossary.

A

Access Control Lists (ACL) A structure attached to a software object that defines access permissions for multiple users and groups. It extends the permissions defined by the HP-UX file system's mode bits by letting you specify the access rights of many individuals and groups instead of just one of each.

Administrative Host See *local host*.

Agent The agent (swagent) runs on the local host. It services all selection, analysis, execution and status requests. It is scheduled by the *daemon* and guided by the SD-UX *controller*.

Alternate Depot Directory A depot directory located someplace other than the default location.

Alternate Root/Alternate Root Directory A Target for software installation, where the Target is not the *primary Root* (/) and where the software can be stored or referenced, but not configured or used.

Analysis/Analysis Phase The second phase of a software installation, copy, or remove operation, during which the host executes a series of checks to determine if the selected products can be installed, copied, removed, or verified on the host. The checks include the execution of check scripts and *disk space analysis* (DSA).

Ancestor An *attribute* that names a previous version of a fileset. This is used to match filesets on a target system. If the `match_target` option is set to true, SD-UX matches the ancestor fileset name to the new fileset name.

Applied The state in which a patch is installed. When a patch is installed, by default it has the `patch_state` of applied. Other patch states include *committed* and *superseded* and *committed/superseded*.

Architecture A keyword that represents the operating system platform on which the product runs.

Archive file A `.o` file that needs to be replaced in an existing archive using the ar command. Used for patch files.

Ask An operation in which SD-UX runs an interactive *request script* to get a response from the user. Request scripts can be run by the swask, swconfig, and swinstall commands.

Attributes Information describing a software object's characteristics. For example, product attributes include revision number, tag (name), and contents (list of filesets). Fileset attributes include tag, revision, kernel, and reboot. File attributes include mode, owner, and group. An essential part of the *Product Specification File*, attributes include such information as the product's short name or tag, a one-line full name title or a one paragraph description of the object. Other attributes include a multi-paragraph README file, a copyright information statement and others.

Authorization In SD-UX security, checking that a user has the necessary permissions to perform a specific action, as defined by an *Access Control List*.

B

Base software Software that will be modified by a patch.

Building phase Packaging the source files and information into a product, and creating/merging the product into the destination depot/media.

Bundles A collection of filesets that are encapsulated for a specific purpose. By specifying a bundle, all products or filesets under that bundle are automatically included in the operation.

C

Cache File A file that contains the name and attributes of targets selected by swinstall or swcopy.

Catalog/Catalog directory An area within a depot that contains all the information needed by SD-UX to define the organization and contents of the products stored in the depot. It includes a global INDEX file and a directory of information for each product version in the depot. It is sometimes referred to as the catalog directory.

Category This keyword defines the “category” attribute for the product object. It refers to the type of software being packaged.

CD-ROM Compact Disc-Read Only Memory or a SD-UX depot that resides on a CD-ROM.

Centralized management See *remote operations*.

Checkinstall script An optional, script associated with a product or fileset, executed by swinstall during the analysis phase. The result returned by the script determines if the fileset can be installed or updated.

Checkremove script An optional script associated with a fileset that is executed during the swremove analysis phase. The result returned by the script determines if the fileset can be removed.

checksum Cyclic Redundancy Check (CRC), a computed value that is compared with stored data to tell if a file has been corrupted during transfer.

CLI Command Line Interface. See *Command Line User Interface*.

Client Usually refers to diskless server computer. Previous versions of SD-UX supported diskless clients.

CLUI See *Command Line User Interface*. All SD-UX commands can be run from the command line. *See also GUI, TUI*, and *IUI*.

Codeword *See To protect software from unauthorized installation, HP (and other vendors) use special codewords and customer identification numbers to lock the software to a particular owner. These codewords and customer IDs are provided to you when you purchase the software or receive it as update.*

Command line options Optional parameters for a command entered with the command itself at the HP-UX command line prompt. See also *default options*.

Command Line User Interface (CLI/CLUI) Text-formatted commands and options entered at an HP-UX command line prompt or executed by a script. SD-UX also has a *Graphical User Interface (GUI)* and a *Terminal User Interface (TUI)* for the sd, swinstall, swcopy, swlist, and swremove commands.

Committed The state in which a patch is applied and rollback files have been deleted. Other patch states include *applied* and *superseded* and *committed/superseded*.

Committed/superseded A patch state in which the patch is both *committed* and *superseded*.

Compatibility Filtering The ability of swinstall to filter the software available from a source according to the host's uname attributes. Software products are created to run on specific computer hardware and operating systems. Many versions of the same products may exist, each of which runs on a different combination of computer hardware and operating system. By default, swinstall does not allow selection and installation of incompatible software.

Compatible Software A software product that will operate on a given hardware system. Software that passes *compatibility filtering* for a local host. Also see *Incompatible Software*.

Configure Script An optional script associated with a fileset and automatically executed by swinstall (or manually executed by swconfig) after the installation of filesets is complete.

Container ACL Template A special ACL (`global_soc_template`) that is used to create initial ACLs for depot and roots. See also *product ACL template*.

Contents A keyword used to assign filesets to subproducts. This allows a fileset to be contained in multiple subproducts.

Controller The SD-UX programs or commands (swinstall, swcopy, etc.) that are invoked by the user on the local host and that direct the actions of an SD-UX *agent*.

Control Script Optional scripts packaged with software or added to software by modifying the IPD. Control scripts are run during swconfig, swinstall, swremove, or swverify operations. Control scripts may include: configure or unconfigure for swconfig; checkinstall, preinstall, postinstall and configure scripts for swinstall; the checkremove, unconfigure, preremove, and postremove scripts for swremove; and the fix or verify script for swverify.

Copyright A keyword that defines the copyright attribute for the destination depot (media) being created/modified by swpackage. It refers to the copyright information for the software product.

Corequisite A *dependency* in which a fileset requires that another fileset be installed or configured at the same time. For example, if fileset A requires that fileset B is installed at the same time, fileset B is a corequisite.

Critical Fileset A fileset containing software critical to the correct operation of the host. Critical filesets are those with the reboot and/or kernel fileset flags. During swinstall's load phase, critical filesets are loaded and customized before other filesets.

Cumulative patch See *superseding patch*.

D

Daemon The SD-UX program that schedules the *agent* to perform software management tasks. On a SD-UX *controller*, the daemon polls the job queue for scheduled jobs.

data_model_revision The internal attribute for SD-UX INDEX file syntax. Layout_version 1.0 uses data_model_revision 2.40; whereas, layout_version 0.8 uses data_model_revision 2.10.

DCE Distributed Computing Environment. Technology used by SD-UX for distributed communications. *Controllers*, *daemons*, and *agents* communicate using the DCE *Remote Procedure Call* (RPC).

Default Hosts File The file (either `/var/adm/sw/defaults.hosts` for system level defaults) or `/$HOME/.sw/defaults.hosts` for user level defaults) that contains the default list of *hosts* for SD-UX commands.

Defaults File The file (either `/var/adm/sw/defaults` for system-wide defaults or `$HOME/.sw/defaults` for user- level defaults), which contains the *default options* and operands for each SD-UX command.

Default Options Changeable values that affect SD-UX command behaviors and policies. Default options are contained in the *defaults file*. See Appendix A, "Command Options," on page 421 for more information.

Delegation SD-UX provides a controlled access to depot-resident products: both the *host* where the *agent* is running and the user initiating the call (delegation) must have read access.

Dependency A relationship between fileset in which one requires another in a specific manner. For example, before fileset A can be installed, it may require fileset B to be installed. SD-UX supports *corequisite*, *exrequisite*, and *prerequisite* dependencies. See Dependent.

Dependent A fileset that has a *dependency* on another fileset. For example, if fileset A depends on fileset B, then B is a dependent or has a dependency on A.

Depot A repository of software products and a *catalog,* organized so SD-UX commands can use it as a *software source*. The contents of a depot reside in a directory structure with a single, common root. A depot can exist as a directory tree on a SD-UX file system or on CD-ROM media, and it can exist as a tar archive on a serial media (tape). All depots share a single logical format, independent of the type of media on which the depot resides. Depots can reside on a local or remote system. You can package software directly into a depot or copy packaged software into the depot from elsewhere.

Depot Source See *depot*.

Destination The path at which a file will be installed.

Developer Host A system where software application files are placed for further integration and preparation for distribution. You may use a developer host to assemble, organize, and create product tapes or depots.

Description An attribute for products and filesets, usually a paragraph description of that product or fileset.

Details Dialog In the *GUI* or *TUI*, a dialog box that lets you get more information about a specific process to monitor its progress.

Directory In packaging, a keyword that defines the a directory for a product object. The directory specified is a default, absolute pathname to the directory in which the product will be installed.

Directory Depot The directory on a target host where a depot is located. The default is `/var/spool/sw`.

Disk Space Analysis (DSA) A process that determines if a host's available disk space is sufficient for the selected products to be installed.

Downdating Overwriting an installed version of software with an older version.

DSA See *Disk Space Analysis*

E

End An optional keyword that ends the software object specification in a *PSF*. No value is required.

Exrequisite A *dependency* in which a fileset requires the absence of another fileset before it can be installed or configured. For example, if fileset A cannot be installed or configured if fileset B is already installed, fileset B is an exrequisite for fileset A.

F

Fileset A collection of files. Most SD-UX operations are performed on filesets.

G

Group In SD-UX security, a set of users.

Group Name In SD-UX security, the user's primary group.

Graphical User Interface (GUI) An OSF/Motif ™ user interface, with windows and pull-down menus, provided with the sd, swinstall, swcopy, swlist, and swremove commands. See also the *Command Line User Interface (CLUI)* and *Terminal User Interface (TUI).*

GUI *See Graphical User Interface.*

H

HOME A variable that contains the path of the current user's local log-in directory.

Host A computer system upon which SD-UX operations are performed. *See local host* and *controller*.

Host ACL The ACL that is attached to and controls access to the *host* object.

I

Incompatible Software Software products are created to run on specific computer hardware and operating systems. Many versions of the same products may exist, each of which runs on a different combination of hardware and operating system. Incompatible software does not operate on the host(s) because of the host’s computer hardware or operating system. The default condition in swinstall is to disallow selection and installation of incompatible software.

INDEX/INDEX file In packaging, an INDEX file defines attribute and organizational information about an object (for example, depot, product, or fileset). INDEX files exist in the depot catalog and the *Installed Products Database* to describe their contents.

INFO An INFO file provides information about the files contained within a fileset. This information includes type, mode, ownership, checksum, size, and pathname attributes. INFO files exist in the depot catalog and the *Installed Products Database* to describe the files contained in each existing fileset.

Input Files Defaults files, option files, software selection files, target host files, and session files that modify and control the behavior of the SD-UX commands.

install-sd A command that lets you install the SD-UX product from media or a depot onto a workstation or server. You may need to install SD-UX if the version on your system is corrupted or deleted. This command, along with *update-ux*, replaces the older *swgettools* command.

Installed Product A product that has been installed on a host so that its files can be used by end-users, as opposed to a product residing in a depot on a host’s file system. Sometimes referred to as an available product.

Installed Products Database (IPD)

Describes the products that are installed on any given host (or within an alternate root). Installed product information is created by swinstall, and managed by swmodify. The contents of an IPD reside in a directory structure with a single common root.

Instance_ID A product attribute in the *Installed Products Database* (IPD) that lets you uniquely identify products with the same tag (name) or revision.

IPD *See See Installed Products Database.*

Is_Locatable In packaging, a keyword that defines whether a product can be installed to an alternate product directory or not. If specified, the attribute is set to a value of true. If not specified, the attribute is assigned a value of false.

IUI Interactive User Interface, a generic term that can mean either the *Graphical User Interface (GUI)* or the *Terminal User Interface (TUI)*.

J

Job A SD-UX task created by the swinstall, swcopy, swremove, swverify, or swconfig commands. You create, monitor, schedule, and delete jobs using the *Job Browser*. You can also monitor jobs using the *swjob* command.

Job Browser A *GUI* program that lets you create, monitor, schedule, and delete jobs. The GUI is activated by the *sd* command. You can also monitor jobs using the *swjob* command.

Job ID Unique numbers generated by SD-UX to identify jobs.

K

Kernel Fileset A fileset that contains files used to generate the operating system kernel. During the swinstall load phase, kernel filesets are loaded and customized before other filesets.

Keyword In packaging, a word (or statement) that tells swpackage about the structure or content of the software objects being packaged by the user. Packaging information is input to swpackage using a *Product Specification File*.

L

Load/Load Phase The third phase of a software installation or copy operation; when swinstall and swcopy load product files on to the host; and when swinstall performs product-specific customization.

Local Host The host on which SD-UX commands are being executed. Sometimes called the administrative host. The local host executes the *controller*, which may direct operations on multiple remote systems when *remote operations* are enabled.

Locatable Product A product that can be relocated to an alternate product directory when it is installed. If a product is not locatable, then it must always be installed within the defined product directory.

Logging Each SD-UX command records its actions in log files (the swlist command is an exception). The default location for the various log files is `/var/adm/sw/<command>.log`

M

Machine_Type In packaging, a keyword that type of systems on which the product will run. (If not specified, the keyword is assigned a wildcard value of * (meaning it will run on all machines.) If there are multiple machine platforms, you must separate each machine designation with a `|` (vertical bar).

Make Tape Phase In packaging software to a distribution tape, this phase actually copies the contents of the temporary depot to the tape.

Media Physical data storage media on which software is stored, such as tape, CD-ROM, or DVD.

Minfree Minimum Free Threshold, the minimum amount of free disk space required to store products being packaged.

multi_stream *See See multiple architecture.*

Multiple Architecture A single product that contains different versions of the same fileset.

N

Network Source There can be multiple network sources from a single host, each one a different depot served by that host's single swagentd daemon. A network source is identified by the host name and depot directory.

Nodes Another name for client host. *See Client*.

Number In packaging, a keyword that defines the part or manufacturing number of the distribution media (CD or tape depot).

O

Object The pieces of software that SD-UX packages, distributes, installs, and manages. There are three classes of objects: software (installed on target roots or available in depots), containers (depot, roots, alternate roots), and *jobs*.

OS Operating System.

owner An attribute indicating the owner of the file (string).

P

Package Installable SD-UX format software created with *swpackage*. Packaged software can be placed in a *depot* for distribution.

Packager The *swpackage* program, which packages software for later distribution to Target systems.

Packaging The task of creating a *package*.

Package Building Phase A phase where swpackage builds source files and information into a product object, and inserts the product into an existing *depot*. If the depot does not exist, swpackage creates a new depot but does not register it.

Package Selection Phase In packaging, reading the *`product_specification_file`* to determine the product, subproduct and fileset structure; the files contained in each fileset; and the attributes associated with these objects.

Patch Software designed to update specific bundles, products, subproducts, filesets, or files on your system. There are *`point`* patches and *`superseding`* (cumulative) patches. By definition, patch software is packaged with the `is_patch` attribute set to true.

Path An attribute that specifies the full pathname for a file.

Point patches Patches that patch separate parts of the same base fileset.

POSIX POSIX 1387.2-1995 IEEE standard, on which SD-UX is based.

Postinstall Script An optional, script associated with a fileset that is executed by swinstall after the corresponding fileset has been installed or updated.

Postremove Script An optional, script associated with a fileset that is executed by swremove after the corresponding fileset has been removed.

Prerequisite A *dependency* in which one fileset requires another fileset to be installed or configured before the first fileset can be installed or configured. For example, fileset A may require that fileset B is installed before fileset A can be installed. Therefore, fileset B is a prerequisite for fileset A. See *dependency*, *corequisite*, and *exrequisite*.

Preinstall Script An optional, script associated with a fileset that is executed by swinstall before installing or updating the fileset.

Preremove Script An optional, script associated with a fileset that is executed by swremove before removing the fileset.

Primary Root A system on which software is installed and configured.

Principal In SD-UX security, the user (or host system, for agents making RPCs) that originates a call to another system.

Product A collection of subproducts and/or filesets.

Product ACL Template In SD-UX security, the ACL used to initialize the ACLs that protect new products on depots that are created by the host.

Product Directory The root directory of a product object, in which most of its files are contained. You can change (relocate) the default product directory when you installing a *locatable product*.

Product Specification File (PSF) An input file that defines the structure and attributes of the files to be packaged by *swpackage*.

Product Version A depot can contain multiple versions of a product. Product versions have the same tag attribute, but different version attributes. *See Multiple Version*. The *installed products database* supports multiple installed versions of a product. Installed versions have the same tag attribute, but different version attributes or a different product directory.

Protected software Software that you cannot install or copy unless you provide a codeword and customer ID. (These are found on your software certificate in your media kit.) You can use codeword-protected software only on systems that for which you have a valid license to use that software.

PSF *See See Product Specification File.*

Pull Getting software products from a *depot* to be installed or copied onto the local system. See also *push*.

Push Performing software management (usually installing or copying) on multiple remote *target* systems from a central *controller*. See *remote operations*.

R

Readme This keyword defines the "readme" attribute for the product object. A text file of the README information for the product; either the text value itself or a file name that contains the text.

Realm In SD-UX security, the scope of the authority by which the *principal* is authenticated.

Register/Registration A process that determines what depots are available on a given host and makes them available for use. Registration information consists of the depot or root's identifier (its path in the host file system). This information is maintained by the daemon which reads its own file at start-up.

Remote Host A Host other than the one on which the SD-UX commands are being executed.

Remote Operations Performing operations on remote systems from a single controller system. Remote operations must be enabled. (Also called *centralized management* or *single point administration*.) See Chapter 6, "Remote Operations Overview," on page 189 for more information.

Remote Procedure Call (RPC) Refers to the operations with Agents on a remote computer.

Request script An interactive control script that gets a response from the user. A request script prompts the user for a response, reads the user's answer, and stores the results in a *response file*. Request scripts can be run by the swask, swconfig, and swinstall commands.

Response file A file that is generated by an interactive *request script* and contains the user's response.

Revision This keyword defines the "revision" attribute for the product object. The revision information (release number, version) for the product.

Root The root directory of a system (/). *See Root Directory.*

Root Directory The directory on a target host in which all the files of the selected products will be installed. The default (/), can be changed to install into a directory that will eventually act as the root to another system. *See Alternate Root Directory*.

RPC Remote Procedure Call. DCE technology for distributed communications and data transfer.

S

sd The command that invokes the *Job Browser*, a *GUI* program that lets you create, monitor, schedule, and delete jobs. The *swjob* command lets you monitor jobs from the command line. You can also activate the Job Browser with the swjob -i command.

SD format See *SD-UX format*.

SD-UX format The format and syntax of SD-UX software in depots. See *Layout_version.*

Secret In SD-UX security, a password used to verify the authenticity of the caller's host. SD-UX manages sets of hosts by restricting and changing the default secret on all *controller* and *target* hosts in the network. See *shared secrets file*.

Security Controlling access to software objects. In SD-UX, security is achieved by a combination of Access Control Lists (ACLs) associated with objects and commands, and the security inherent in the file system permissions on which the software is stored. See *Access Control List.*

Selection, Selection Phase The first phase of a software installation, copy, remove, or verify operation, during which the user selects the software products to be installed, copied, or removed from the host.

Server A system on the network that acts as a software source for other systems on the network.

ServiceControl Manager (SCM) An HP program that permits central management of many system administration functions. You can run SD-UX from SCM.

Session/Session File Each invocation of a SD-UX command defines a session. Most SD commands let you use the `-C` *`session_file`* option to save command options, source information, software selections, and host selections and re-use this information with the `-S` *`session_file`* option. You can also save and re-use session information from the GUI programs.

Shared Secrets File In SD-UX security, a file containing the passwords used to encrypt and decrypt distributed communications for added security.

Single Point Administration (SPA) The ability to simultaneously distribute to, manage, or monitor multiple remote targets from a single controller system. See *remote management*.

Software depot An *SD-UX format* structure that contains one or more software products that can be installed on other systems or copied to other *depots*.

Software file An input file of previously defined *software selections* to be used as operands for a command. You specify a software file with the `-f software_file` command line option.

Software group A group of software selections read or saved from the GUI programs.

Software object The objects packaged, distributed, installed, or managed by SD-UX. A software object may be a file, fileset, bundle, or product. Most operations are performed on `filesets`.

Software selection A group of software objects that you have selected for an operation. You can save these software selections for later re-use. See *software group*.

Software Selection Window A GUI window that lets you select the software files you want to install, copy, or remove.

Software source A depot used as the source of a swinstall or swcopy operation.

Source See *software source*.

SPA See *Single Point Administration*.

Staging A way of setting up intermediate depots that are local to each group of targets on local area networks. This can reduce the amount of network traffic.

Staged installation See *staging*.

State An attribute that indicates the current state of the fileset. During installation, software is transitioned through the following states: non-existent, *transient*, *installed*, and *configured*. During removal, software is transitioned through these states: *configured*, *installed*, *transient*, and *non-existent*. If a task fails during a transient state, the state is set to *corrupt*.

Subproducts An optional grouping of filesets, used to partition a product that contains many filesets or to offer the user different views of the filesets.

Superseded The state in which a patch was applied but was then replaced by a *superseding patch*. Other patch states include *applied* and *committed*.

Superseding patch A patch that supersedes all previous patches to a given fileset.

SW-DIST A software product that provides all of the SD-UX functionality. SW-DIST is included on your HP-UX 11i media. If SW-DIST is damaged, missing, or corrupted on your system, you cannot install or copy any HP-UX software that is packaged in the *SD-UX format*, including a newer SW-DIST product. You can re-install SD-UX with the *install-sd* command.

swacl A SD-UX command that allows you to modify Access Control List permissions that provide software security.

swadm In SD-UX security, the default user identification group.

swagent The SD-UX *agent* program that makes changes to depots and roots. It is directed by the *controller* and scheduled by the *daemon*

swagentd The SD-UX *daemon* that provides various services, including: initiation of communication between the *controller* and *agent*; serving one or more *depots* to multiple requesting *agents* on remote *hosts*.

swask A SD-UX command that lets you run an interactive *request script* to get a response from the user. Request scripts can also be run by the swconfig and swinstall commands.

swconfig A SD-UX command that configures previously installed software and make the software ready for use.

swcopy A SD-UX command that copies software from a software source to a depot or from one depot to another. The swcopy command can add products to an existing depot, replace products already on a depot, or create a new depot.

swgettools A SD-UX command used in previous HP-UX releases to install the new SW-DIST product from media. This command has been replaced by *install-sd* and *update-ux*.

swinstall A SD-UX command that installs software. swinstall may also perform software configuration.

swlist A SD-UX command that lists software objects, their attributes, and their organization. It lists both installed software and software contained within a depot.

swlock A file that contains the read or write access to software objects and ACLs.

swmodify A SD-UX command that lets you change information in the *installed products database* or *depot* catalog files.

swpackage A SD-UX command that uses a *product specification file* (PSF) to organize software products and *package* them into a *depot*. The depot can be accessed directly by SD-UX commands or mastered onto CD-ROM or tape.

swreg A SD-UX command used to *register* or *unregister* depots.

swremove A SD-UX command that removes previously installed software or removes packaged software from a depot.

swverify A SD-UX command that verifies installed software or depot software for correctness and completeness.

Systems Computers, either stand-alone or networked to other computers. See *local host*.

T

Tag In packaging, a keyword that defines the distribution tag or software object's name attribute for the destination depot (media).

Tape Depot A software *depot* stored in a tar (tape archive) format. Within the archive, directory and file entries are organized using the same structure as any other *SD-UX* *format* depot.) Tape depots such as cartridge tapes, DAT and 9-track tape are referred to by the file system path to the tape drive's device file.

Tape Media Software media that uses tar to store SD-UX software products and control files. It usually resides on a serial media such as a DDS, cartridge, nine-track, or other tape, though it can also be a regular file that contains the tar archive. Within the tar archive, directory and file entries are organized using the same structure as any other depot.

Tape Source See *tape depot*.

Target Any system on which software is to be installed or managed with SD-UX. There are typically multiple targets on a network, identified by system name, network address, user name, or by a user group. Targets can contain a *primary root*, an *alternate root*, or *depots*. A target may also be the object of *remote operations*.

Target Group Most SD-UX commands let you use the `-t` *`target_file`* option to read a list of previously defined *target selections* as operands for the command. You can also read or save target group files from the GUI programs when *remote operations* are enabled.

Target Selection A group of systems or software objects that you have selected as *targets* for an operation. You can save these selections for later re-use. See *target group*.

TUI Terminal user interface. A character-based display with windows and pull-down menus that works on ASCII terminals. The TUI uses the keyboard to navigate (no mouse). See also *Command Line User Interface* and *Graphical User Interface*.

TUI See *Terminal User Interface.*

Title A one-line, full name attribute that identifies the product with a title.

U

UDP/IP User Datagram Protocol. Comparable with TCP/IP, but runs connections less and is intended to be used in more reliable network environments (LAN).

Uname Attribute When a target is contacted for a software management operation, the system's four uname attributes (operating system name, release, version and hardware machine type) are obtained. Used to determine software compatibility with the proposed host.

Unconfigure Script An optional script that undoes the configuration done by the configure script. Unconfigure scripts are associated with filesets and are automatically executed by swremove before the removal of filesets begins. You can also run unconfigure scripts with swconfig.

Unregister Using the *swreg* command to remove the *registration* of a *depot*. This makes the depot unavailable to network access.

Update Overwriting software objects already installed on the system and replacing them with new objects.

update-ux A command that automates part of the HP-UX update process. It replaces the *swgettools* script used in previous versions of SD-UX. The *install-sd* updates the SD-UX product without performing an OS update.

User name The user (or host system for agents making remote procedure calls (RPCs) to other agents) that is originating the RPC call.

UUID In packaging, a keyword that for the vendor object. Useful for NetLS vendors and for those who want to select products from two vendors who have chosen the same `vendor_tag`.

V - Z

Vendor If a vendor specification is included in the PSF, swpackage requires the vendor and tag keywords.

Vendor_tag Associates the product or bundle with the last-defined vendor object, if that object has a matching *tag* attribute.

Verbose Listing A listing that is used to display all attributes for products, subproducts, filesets, or files.

Index

Index

Index

Index