



HEWLETT
PACKARD

Getting Started with the
Domain Software Engineering
Environment (DSEE)

Getting Started with the
Domain Software
Engineering Environment
(DSEE)

Order No. 008788-A01



© Hewlett-Packard Co. 1986, 1988, 1991.

First Printing: March 1986

Last Printing: April 1991

UNIX is a registered trademark of UNIX System Laboratories Inc.

X Window System is a trademark of the Massachusetts Institute of Technology.

OSF/Motif is a trademark of the Open Software Foundation, Inc. in the USA and other countries.

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

* This document contains proprietary information which is protected by copyright. All rights reserved. No part of this document may be photocopied, reproduced or translated to another language without the prior written consent of Hewlett-Packard Company.

RESTRICTED RIGHTS LEGEND. Use, duplication, or disclosure by government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227.7013. Hewlett-Packard Co., 3000 Hanover St., Palo Alto, CA 94304

10 9 8 7 6 5 4 3 2 1

Preface

Getting Started with the Domain Software Engineering Environment (DSEE) introduces the basic concepts and commands of the DSEE facility. This manual also contains a sample DSEE session that demonstrates how new DSEE users can become productive quickly on projects currently managed by the DSEE facility.

The intended audience for this manual includes software engineers who are already familiar with the Domain/OS system as described in *Getting Started with Domain/OS*.

The Organization of This Manual

This manual includes a sample DSEE session that illustrates how you can manage sources, build a program, create a release, and manage a project with the DSEE facility.

- | | |
|------------------|---|
| Chapter 1 | Briefly describes the DSEE facility, the DSEE desktop interface, the sample program used in all the examples, and a few miscellaneous commands. |
| Chapter 2 | Describes how to manage sources in the DSEE facility using libraries, elements, and branches. |

Chapter 3	Explains a simple build procedure in the DSEE facility and defines a system, a system model, and a configuration thread.
Chapter 4	Discusses the rebuilding of a program after modifying its sources, system model, and configuration thread. This chapter also demonstrates the use of equivalences and concurrent building on multiple nodes.
Chapter 5	Describes the release of an executable program.
Chapter 6	Explains the use of tasks, tasklists, and monitors to aid project management.
Glossary	Defines DSEE terminology.

Requirements

Because this manual makes frequent references to online information, we recommend that you read it while sitting at a node. The node must have the following software:

- Domain Software Engineering Environment (DSEE), Version 4.0 or higher
- Domain/C compiler

In addition, the node needs to have the following programs running as server processes:

- `d3m_server`
- `mbx_helper`
- `Xapollo`

If you're not sure whether these programs are running, issue the UNIX shell command `/bin/ps -ax` (BSD) or `ps -e` (SysV), or the Aegis shell command `/com/pst`. For information on installing DSEE software, contact your system administrator or read the *DSEE Software Release Document*.

Summary of Technical Changes

Getting Started with DSEE documents technical changes to the DSEE facility that have been made since its last release, Version 3.0. We have revised *Getting Started with DSEE* to show how to issue commands using the new graphical interface based on the X Window System and OSF/Motif. In addition, Chapter 1 introduces the graphical interface.

For details about the technical changes to the DSEE facility, refer to the *DSEE Release Document*.

Related Manuals

The file `/install/doc/apollo/os.v.latest software release number__manuals` lists current titles and revisions for all available manuals.

For example, at SR10.3 refer to `/install/doc/apollo/os.v.10.3__manuals` to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

(If you are using the Aegis environment, you can access the same information through the Help system by typing `help manuals`.)

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete

list of related documents. For more information about the Domain system, refer to the following documents:

- *Getting Started with Domain/OS* (002348) describes the basics of the Domain/OS operating system.
- *Using your Aegis Environment* (0011021) is a detailed guide to using the Aegis environment.
- *Using your BSD Environment* (0011020) is a detailed guide to using the BSD environment.
- *Using your SysV Environment* (0011022) is a detailed guide to using the SysV environment.
- *Aegis Command Reference* (002547) provides detailed information about the Aegis shell.
- *BSD Command Reference* (005800) describes the BSD shell commands supported by Domain/OS.
- *SysV Command Reference* (005798) describes the SysV shell commands supported by Domain/OS.

For more information about the X Window System and how to configure X resources, refer to the *X Window System User's Guide* [O'Reilly Vol. III] (015534).

For more information about the DSEE facility, refer to the following documents:

- *Using the Domain Software Engineering Environment (DSEE)* (015798) describes the concepts behind the DSEE facility, explains how to use DSEE functions, and provides case studies that illustrate DSEE features. We refer to this manual as *Using DSEE* throughout this book.
- *Domain Software Engineering Environment (DSEE) Reference* (003016) provides detailed information on DSEE

commands and the DSEE callable interface. We refer to this manual as the *DSEE Reference* throughout this book.

You can order Apollo documentation by calling **1-800-225-5290**. If you are calling from outside the U.S., you can dial **(508) 256-6600** and ask for **Apollo Direct Channel**.

Does This Manual Support Your Software?

This manual was released with software version 4.0 of the Domain Software Engineering Environment. Version 4.0 runs on SR10.2, or a later release of Domain/OS. To verify which version of operating system software you are running, type:

`bldt`

To check the version of DSEE software, type:

`/com/dsee -version`

If you are using a later version of software than that with which this manual was released, use one of the following ways to check if this manual was revised or if additional manuals exist:

- Read Chapter 3 of the release document that shipped with your product. The release document is online: `/install/doc/apollo/dsee.v.4.0__notes`. Check with your system administrator if you cannot find the release document.
- Telephone **1-800-225-5290**. If you are calling from outside the U.S., dial **(508) 256-6600** and ask for **Apollo Direct Channel**.
- Refer to the lists of manuals described in the preceding section, "Related Manuals."

To determine which of two versions of the same manual is newer, refer to the order number that is printed on the title page. Every order number has a 3-digit suffix; for example, **-A00**. A higher suffix number indicates a more recently released manual. For example, a manual with suffix **-A02** is newer than the same manual with suffix **-A01**.

Problems, Questions, and Suggestions

If you have any questions or problems with our hardware, software, or documentation, please contact either your HP Response Center or your local HP representative.

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about documentation.

Documentation Conventions

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values	Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.
<i>user-supplied values</i>	Italic words or characters in formats and command descriptions represent values that you must supply.
sample user input	In interactive examples, information that the user enters appears in color.
output/source code	Information that the system displays appears in this typeface. Examples of source code also appear in this typeface.

[] Square brackets enclose optional items in formats and command descriptions.

{ } Braces enclose a list from which you must choose an item in formats and command descriptions.

| A vertical bar separates items in a list of choices.

< > Angle brackets enclose the name of a key on the keyboard.

CTRL/
^ The notation CTRL/ or ^ followed by the name of a key indicates a control character sequence. Hold down <CTRL> while you press the key.

. . . Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

.
.
.
 Vertical ellipsis points mean that irrelevant parts of a figure or example have been omitted.

————— ☐ ————— This symbol indicates the end of a chapter or part of a manual.

————— ☐ —————

Contents

Chapter 1 Introduction

Capabilities	1-2
Overview	1-2
Before You Begin	1-6
Invoking the DSEE Facility	1-7
Using the Desktop Interface	1-8
Transcript Area Menu	1-15
Sample DSEE Session	1-16
DSEE Documentation	1-17
Exiting the DSEE Facility	1-18

Chapter 2 Managing Sources

Using Libraries	2-3
Setting Contexts	2-3
Exploring the Contents of a Library	2-6
Listing the Contents of a Library	2-10
Creating a Library	2-11
Using Elements	2-12
Examining Elements	2-13
Examining Element Histories	2-14
Modifying Elements	2-15
Creating Elements	2-20

Creating Branches	2-22
Merging Branches	2-25
Naming Element Versions	2-32
Deleting Elements	2-34
Removing Icons	2-34
Command Summary	2-34
Related Information	2-37

Chapter 3 Building a Program

Preparing for a Build	3-2
The System	3-2
The System Model	3-3
Block Types	3-5
Declarations	3-6
Setting the Current System Model	3-8
The Configuration Thread	3-10
The Model Thread	3-14
Initiating a Build	3-15
What Happens During a Build	3-18
Examining a Build	3-20
Accessing Derived Objects	3-23
Related Information	3-28

Chapter 4 Modifying and Rebuilding a Program

Modifying a Program	4-2
Modifying the System Model	4-4
Performing a Partial Build	4-7
Building a Single Component	4-7
Debugging Source Elements	4-9
Rebuilding the Program	4-11
Building on Several Nodes Concurrently	4-11
Rebuilding with Reserved Elements	4-14
Testing the Program	4-15
Promoting Derived Objects in a Reserved Pool	4-16
Naming Versions Used in a Build	4-17

Rebuilding with Existing Components	4-21
Identifying Components Targeted for Rebuilding	4-22
Building BCTs Only	4-24
Comparing Builds	4-25
Specifying Equivalences	4-26
Rebuilding an Earlier Version of the Program	4-31
Editing the Configuration Thread	4-32
Rebuilding the Program	4-34
Command Summary	4-36
Related Information	4-37

Chapter 5 Managing Releases

Creating a Release	5-1
Command Summary	5-5
Related Information	5-6

Chapter 6 Managing a Project

Using Tasks and Tasklists	6-2
Creating a Task	6-3
Editing a Task	6-5
Examining Tasklists	6-9
Recording Events in Tasks	6-10
Using Monitors	6-16
Creating a Monitor	6-16
Activating a Monitor	6-20
Deleting a Monitor	6-24
Command Summary	6-24
Related Information	6-25

Figures

Figure 1-1.	Representation of a Library	1-3
Figure 1-2.	Representation of a Tasklist and Tasks ..	1-4
Figure 1-3.	Representation of the Build Process	1-5
Figure 1-4.	The DSEE Desktop Interface	1-7
Figure 1-5.	Selecting Commands from the Help Menu	1-9
Figure 1-6.	help Command Dialog Box	1-10
Figure 1-7.	Scroll Bars for the Transcript Area	1-11
Figure 1-8.	Vertical Scroll Bar	1-12
Figure 1-9.	Help Window for reserve Command	1-13
Figure 1-10.	Selecting set shell	1-14
Figure 1-11.	Transcript Area Menu	1-15
Figure 2-1.	Selecting cd	2-4
Figure 2-2.	The cd Dialog Box	2-4
Figure 2-3.	Current Library and Directory Settings ..	2-5
Figure 2-4.	Browsing the Current Library	2-7
Figure 2-5.	Icons on the Desktop	2-9
Figure 2-6.	Selecting show elements	2-10
Figure 2-7.	show elements -full Command	2-11
Figure 2-8.	Selecting the show history Command ...	2-14
Figure 2-9.	The Branch Icon for test	2-17
Figure 2-10.	The File Icon for test	2-17
Figure 2-11.	The compare Dialog Box	2-19
Figure 2-12.	reserve , replace , and compare Commands	2-20
Figure 2-13.	create element Command	2-21
Figure 2-14.	Representation of Element test with Branch	2-24
Figure 2-15.	Representation of a Merger	2-26
Figure 2-16.	Query Box for merge Command	2-28
Figure 2-17.	Restarting the Merger Process	2-30
Figure 2-18.	show derivation Window	2-31
Figure 2-19.	name version Query Box	2-33

Figure 3-1.	set system Command	3-3
Figure 3-2.	A System Model Block	3-4
Figure 3-3.	Block Structure of a System Model	3-5
Figure 3-4.	set model Command	3-9
Figure 3-5.	set thread Command	3-12
Figure 3-6.	build Command	3-16
Figure 3-7.	build Command	3-17
Figure 3-8.	Flow of Information During a Build	3-19
Figure 3-9.	show builds Command	3-21
Figure 3-10.	examine build Command	3-22
Figure 3-11.	Browse Box for Build Units of System Model	3-24
Figure 3-12.	Build Unit Icon	3-24
Figure 3-13.	Selecting display all builds	3-25
Figure 3-14.	Build Icon	3-25
Figure 4-1.	Unsuccessful Build of Single Component .	4-8
Figure 4-2.	Successful Build of Single Component ...	4-10
Figure 4-3.	DSEE Desktop During Concurrent Build .	4-13
Figure 4-4.	Rebuild of Entire Program	4-15
Figure 4-5.	Build Unit Icon	4-17
Figure 4-6.	Selecting display all builds	4-18
Figure 4-7.	name version Command and Build	4-19
Figure 4-8.	Sample Libraries and Their Element Versions	4-20
Figure 4-9.	Canceled build -query Command	4-23
Figure 4-10.	build -bct_only Command	4-24
Figure 4-11.	Selecting display all builds	4-25
Figure 4-12.	build -query Command with Equivalences	4-30
Figure 4-13.	Sample Libraries and Their Element Versions	4-34
Figure 4-14.	Build without -dbs Translation Option ..	4-35
Figure 5-1.	create release Command	5-3
Figure 5-2.	Structure of Release Directory order_rls .	5-4
Figure 6-1.	Creating a Task	6-5
Figure 6-2.	Task Editor	6-6
Figure 6-3.	Editing a Task	6-8

Figure 6-4. Examining a Tasklist	6-10
Figure 6-5. Checking off an Active Item	6-13
Figure 6-6. A Task Is Noted in Element History	6-14
Figure 6-7. Automatic Cross-Referencing	6-15
Figure 6-8. A Monitor's Activation List	6-19
Figure 6-9. Activating a Monitor	6-22
Figure 6-10. An Alarm Window	6-23
Figure 6-11. A Tasklist with a New Task	6-23

Glossary

Index



Prerequisites to Chapter 1

Before you perform any of the exercises in this chapter, please make sure that you have

- Installed DSEE software, Version 4.0 or higher, on your node.
- Installed an up-to-date version of the Domain/C compiler on your node.

Chapter 1

Introduction

Large-scale software development efforts typically involve many engineers working on different components of the same system. At times, several engineers might even work on the same component simultaneously. In such circumstances, the members of a development team often adopt common coding conventions and establish formal channels of communication to coordinate their efforts. Despite the best of attempts, however, significant and avoidable complications frequently arise.

The Domain Software Engineering Environment (DSEE) provides a structured but flexible environment for software development. Besides handling many of the mundane chores that you seldom have time for and would rather avoid, the DSEE facility also streamlines many important functions. Consequently, you have more time to devote to program development.

This chapter presents an overview of some basic DSEE concepts, introduces the DSEE desktop interface, and describes the sample program used in all the examples presented in this book.

Capabilities

The DSEE facility helps you to:

- Archive previous versions of sources and other text files
- Control access to all versions of these files
- Document the history of each file
- Accommodate alternative versions of files
- Organize work assignments
- Monitor changes to specified files
- Build programs and individual components
- Rebuild programs with previously built components
- Distribute builds over many nodes
- Concurrently build program components
- Manage program releases

Overview

The DSEE facility helps to manage and control access to text files by storing them as elements in a library.

An **element** differs from a text file in several respects. You cannot edit an element directly; instead, you edit a copy of the element. The DSEE facility then uses the edited copy to create a new version of the element. By repeating this process with the latest version of the element, you create a sequence of versions, or generations, called the element's main line of descent. If you wish, you can also create an alternate line of descent (called a branch) for the element.

A **library** is a DSEE database that contains elements normally related by function or form, such as a program's source files, include files, or chapters of documentation. A software development project may use several DSEE libraries. Every library has its own set of protection attributes for controlling access to the library and its contents. (Figure 1-1 illustrates a DSEE library.)

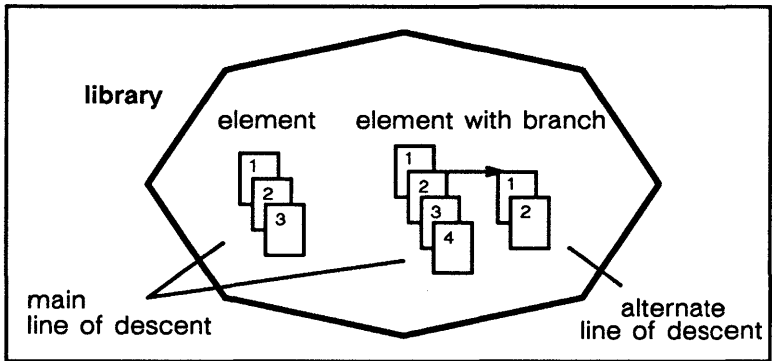


Figure 1-1. Representation of a Library

The DSEE facility helps to organize and coordinate a software development effort with tasks, tasklists, monitors, and forms.

A **task** is a multi-step process that you define. For instance, you might define the task "Fix bug 122," which has the steps "Locate error," "Fix code," and "Release fixed code." Tasks act as both "to do" lists and as records of things that have been done. The DSEE facility maintains a transcript inside the task of the completed items, adding information about elements and libraries impacted by your progress through the task. If several people use the same task, then the task's transcript records their combined efforts.

A **tasklist** helps you to organize and refer to tasks, as illustrated in Figure 1-2. The DSEE facility automatically creates a tasklist for every user and two for every library. You can create additional tasklists in any directory.

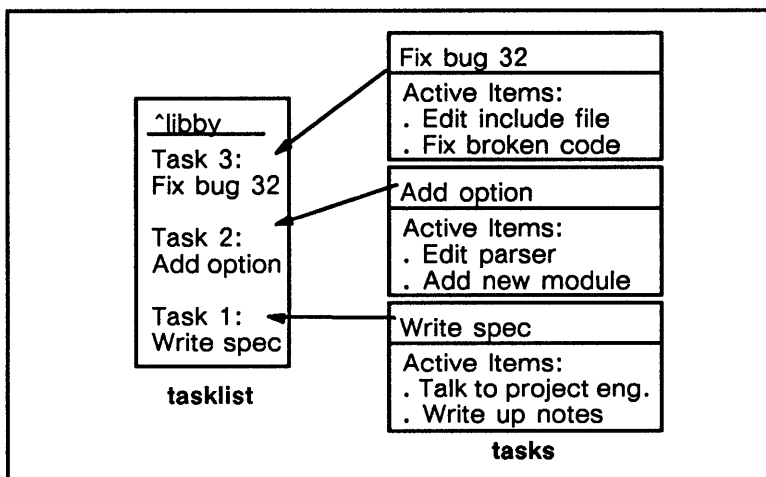


Figure 1-2. Representation of a Tasklist and Tasks

A **monitor** “watches” one or more elements. When someone modifies or deletes a monitored element, the DSEE facility responds by alerting the users and/or executing the shell commands specified by the monitor’s creator.

A **form** typically contains a description of a standard procedure. You can use a form as a template for creating tasks.

In addition to managing sources and improving communications, the DSEE facility also helps to automate system building. Prior to building a system for the first time, you:

- Construct a **system model** describing the system’s components, their dependencies (elements, files, and other components), tool requirements, and build procedures
- Write a **configuration thread** specifying build options and particular versions of the elements identified in the system model
- Create a system directory to serve as a database for the build

Except for changes in the system's design or when building different configurations of the system, your system model and configuration thread generally remain the same for successive builds of the system. As for the system directory, you typically create one at the beginning of a project. You use this system directory throughout the life of that project.

Figure 1-3 demonstrates the build process. At the start of a build, the DSEE facility synthesizes a complete build specification for the system from the system model and configuration thread. The DSEE facility then compares the system's specification against the specifications of previously built components stored in a **binary pool** (a storage area in which the DSEE facility places the results of builds). When the specifications of an existing component and a proposed component match, the DSEE facility reuses the existing component; otherwise, it builds the proposed component. Finally, the DSEE facility stores the results of the build in the binary pool.

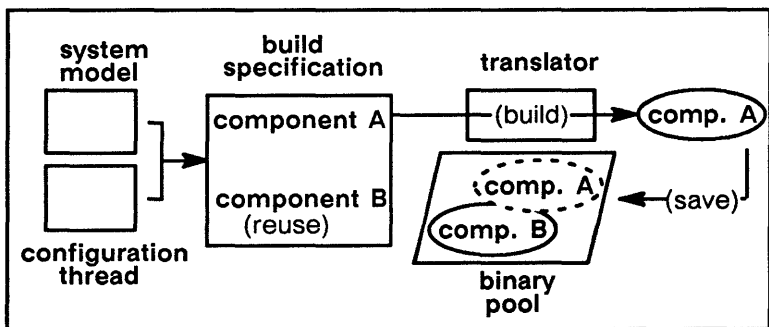


Figure 1-3. Representation of the Build Process

This method of building components only when necessary can save significant amounts of time when rebuilding a system. If your modifications affect only a few components, then the DSEE facility rebuilds only those components. You can also direct the DSEE facility not to rebuild specific components when you know them to be functionally equivalent, though not identical, to existing components.

Another time-saving option in the building process is concurrent building. You can identify many nodes to be used to build components. The DSEE facility then determines which components

can be built concurrently and distributes them among your builder nodes.

The DSEE facility is especially useful for building complex systems involving numerous and diverse sources, multiple contributors, different system configurations, and multiple platforms. With the DSEE facility, you can achieve:

- Concurrent system development and maintenance
- Isolated testing and debugging environments
- Conditional processing of a system model to build systems for different target machines, including workstations other than Domain/OS workstations

The DSEE facility also helps to manage the release of a finished system or system component. When you declare the results of a particular build to be released, the DSEE facility copies all specified built components to a release directory where they can be accessed for general use. In addition, the DSEE facility creates a build map for each released binary and stores the build map with the binary in the release directory.

For more information about the DSEE concepts described in this overview, consult *Using DSEE*.

Before You Begin

Before you can use the DSEE facility, you need to have the proper software installed and running on your node. See the “Requirements” section of the Preface for more information.

In order to use the tutorial this manual is based on, you need to install some sample programs. To install the sample programs, enter the following command at the shell prompt:

```
install_dsee_examples
```

The install procedure creates the subdirectory `dsee_examples` in the directory from which you entered the command.

Invoking the DSEE Facility

Invoke the DSEE facility from any shell by typing:

```
dsee
```

By default, the `dsee` command invokes the desktop interface shown in Figure 1-4. (As described in *Using DSEE*, you can opt to work with the command line interface, instead.)

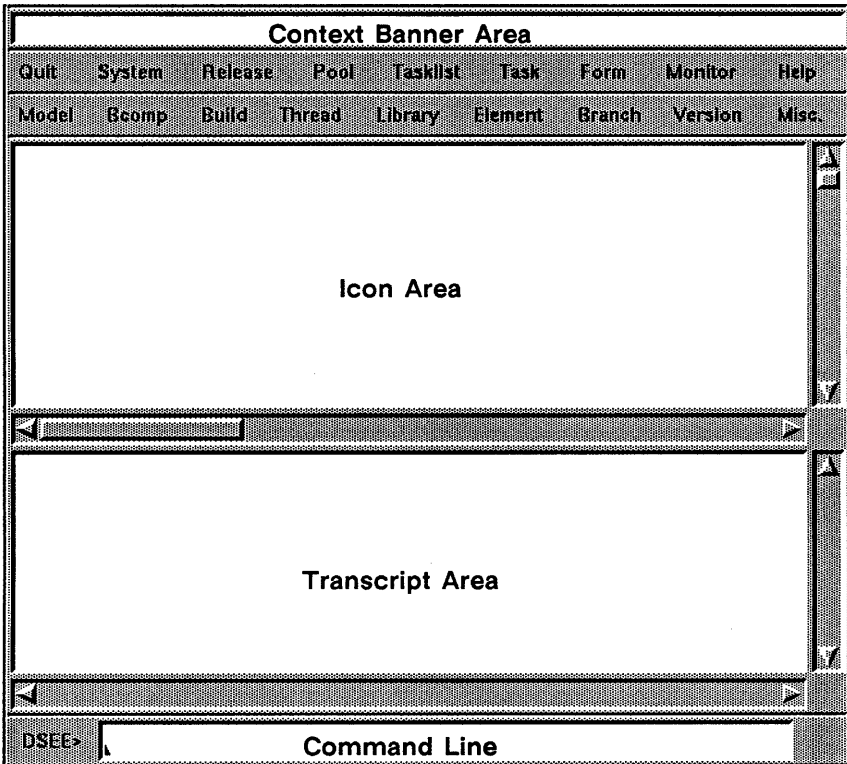


Figure 1-4. The DSEE Desktop Interface

The desktop interface is based on the X Window System and OSF/Motif. You can use X resources to change the appearance of the desktop. (See *DSEE Reference* for details.)

The desktop interface has **context banners** at the top of the window, which indicate your **current settings**. Your current settings are those that were in effect when you last used the DSEE facility. They constitute the working context for DSEE commands executed from the desktop interface. The desktop interface always displays your working directory. Other settings can include your current system, system model, library, tasklist, and task. The DSEE facility requires your current system, system model, and configuration thread settings for a build and requires a library setting when you work with elements.

If you haven't invoked the DSEE facility before, the directory displayed in the context banner area will be the same as the directory from which you invoked the DSEE facility. On subsequent invocations, the working directory will be the one that was in effect the last time you exited from the DSEE facility.

Just below the context banner area are **menu titles** from which you can display menus and issue commands. You can also issue most commands from the command line at the bottom of the desktop interface.

The **icon area** displays icons for different kinds of objects, such as libraries and elements. When you need to enter the names of objects as arguments to DSEE commands, you can often simply click on these icons instead of typing the object's name.

The desktop interface also has a **transcript area**. The DSEE facility uses the transcript area to echo the commands you issue, to display messages, and, where appropriate, to display the results of commands.

Using the Desktop Interface

In this section, we describe how to use some basic features of the DSEE desktop: how to make menu selections, scroll text, copy and paste text, and use pull-down and pop-up menus. We use the **help** command to demonstrate these features.

The **help** command lets you view information on DSEE commands and other subjects. To invoke the **help** command, perform the following steps:

1. Select the pull-down menu for the **help** command: using the mouse, point to the Help menu title and click (press and release) <M'> (the leftmost mouse button).
2. When the menu appears, point to **commands** and click <M1> (see Figure 1-5).

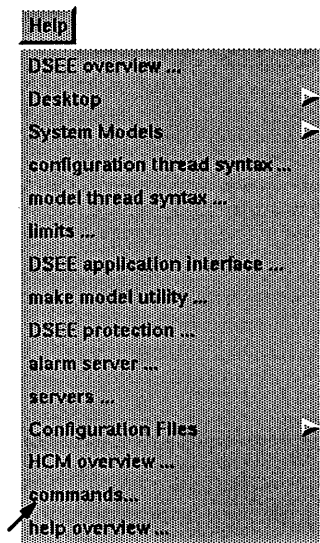


Figure 1-5. Selecting Commands from the Help Menu

Because **commands** is followed by three ellipsis points (...), the DSEE facility displays a **dialog box**, as shown in Figure 1-6. (If a menu entry is, instead, followed by a triangle (▶), the DSEE facility displays a submenu.) A dialog box is a window that appears on your screen. You can use the dialog box to select options and enter command arguments.

To get help on a command, you would enter the command's name in the **text entry box**. The text entry box is the rectangular space with a text cursor in Figure 1-6.

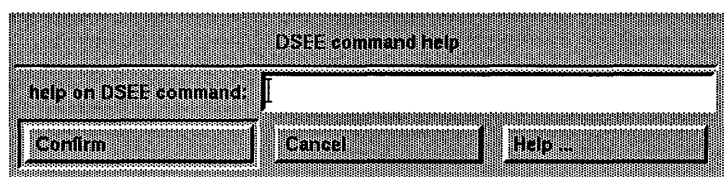


Figure 1-6. help Command Dialog Box

Dialog boxes have buttons. A typical dialog box has a Confirm button, which accepts the information in the dialog box and proceeds with the command, a Cancel button, which closes the dialog box without executing the command, and a Help button, which displays information about the command.

3. Leave the text entry box empty. Point to the Confirm button at the bottom of the dialog box and click <M1>. Because you didn't enter a command name, the DSEE facility displays a list of help topics in the transcript area.

As the DSEE facility lists help topics in the transcript area, many of the topics scroll up and out of view. Use the **scroll bars** to view this information. Scroll bars, shown in Figure 1-7, let you view more text than will fit in the window at a time.

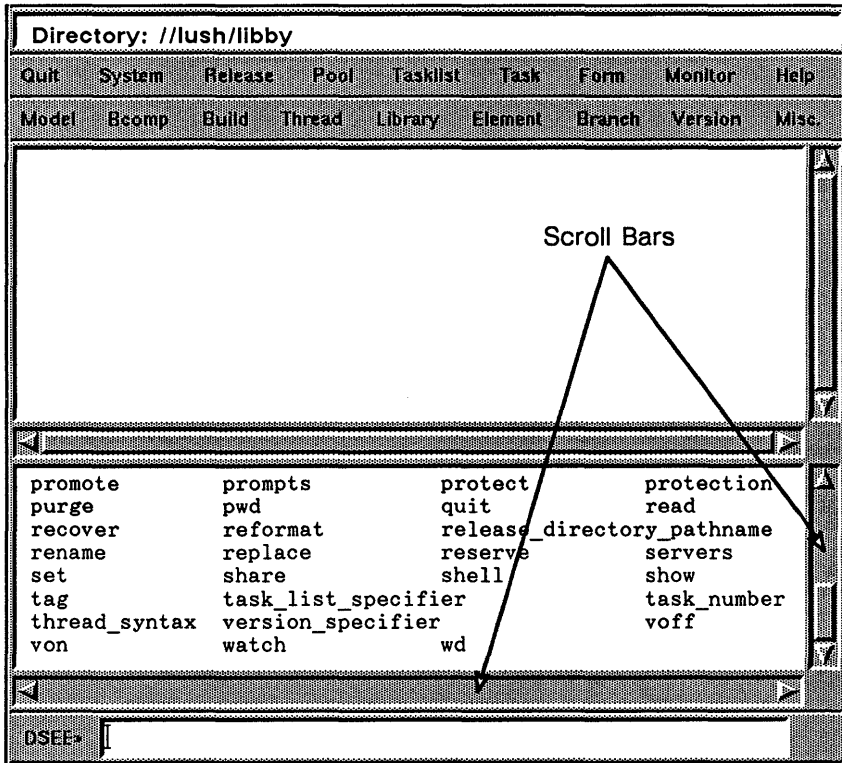


Figure 1-7. Scroll Bars for the Transcript Area

To view hidden text in the transcript area, use the transcript area's vertical scroll bar (see Figure 1-8):

1. Using the mouse, position the cursor over the up-arrow at the top of the transcript area's scroll bar and click <M1> three or four times. Notice that the text moves one line for each click of the mouse button.
2. Point to the down-arrow and click <M1> to move the text one line at a time in the opposite direction.

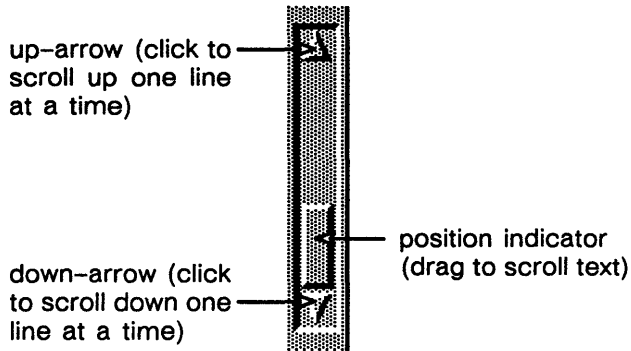


Figure 1-8. Vertical Scroll Bar

To move the text more quickly, move the **position indicator** inside the vertical scroll bar. The position indicator, shown in Figure 1-8, shows the relative position of the text you're viewing with respect to the entire text in the transcript area.

1. Point to the position indicator.
2. Press and hold <M1>. Move the mouse to drag the position indicator up or down. The text scrolls as you move the position indicator.
3. Release <M1>.

You can copy any of the text in the transcript area and paste it into the text entry box of a dialog box, or the input window for command lines. For example, to view a help file for the **reserve** command, try copying **reserve** from the transcript area and pasting it into the **help** dialog box as follows:

1. Select **commands** from the **Help** menu to display the **help** dialog box.
2. Highlight **reserve** in the transcript area: place the cursor in front of the word **reserve**, press and hold <M1>, drag the cursor to the end of **reserve**, and release <M1>.
3. Paste **reserve** in the text entry box of the **help** dialog box: point to the text entry box and click <M2> (the middle mouse button).

4. Point to the Confirm button and click <M1>.
5. The DSEE facility displays a window of information on the **reserve** command (see Figure 1-9). This is the same information you can find if you look up “reserve” in the *DSEE Reference* manual. Use the scroll bars to see this information.

When you are finished viewing the help file, point to the Dismiss button and click <M1>. Dismiss buttons appear on windows that simply display text.

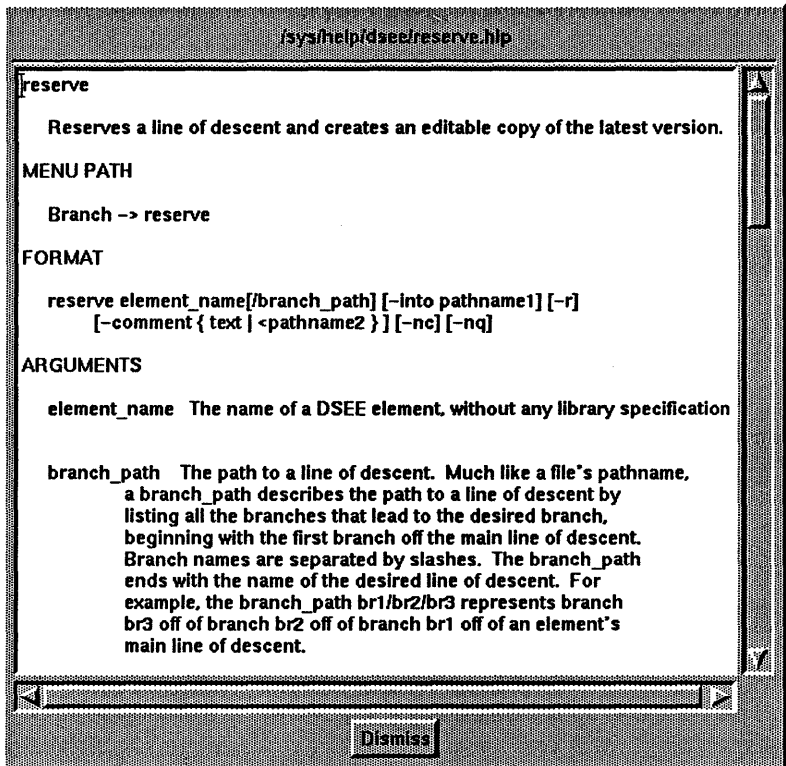


Figure 1-9. Help Window for reserve Command

You can also display help files from the dialog box that appears when you invoke a DSEE command. For example, display the help file for the **set shell** command:

1. Point to the icon area and press and hold <M3>. This displays the pop-up Misc. menu. (You can display other pop-up menus after you place icons in the icon area, as we describe in Chapter 2.)

By moving the mouse while pressing and holding <M3>, you can drag the cursor to select an item instead of pointing and clicking <M3>. Both pull-down and pop-up menus allow you to make selections by pointing and clicking or by dragging the cursor.

2. Select **set shell** from the Misc. menu (see Figure 1-10): slide the cursor off the menu to the right of **Shell**, point to **set shell** on the submenu that appears, then release <M3>.

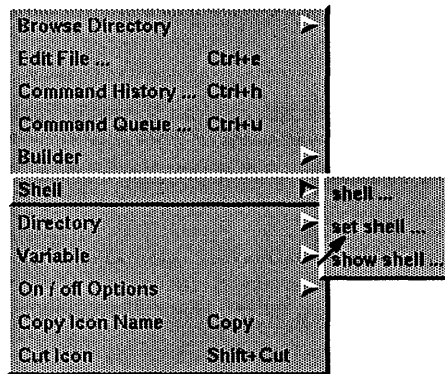


Figure 1-10. Selecting set shell

3. When the dialog box for **set shell** appears, point to the Help button and click <M1>.
4. The DSEE facility displays a window of information on the **set shell** command. When you've finished looking at the information, point to the Dismiss button and click <M1>.
5. Cancel the **set shell** command: point to the Cancel button and click <M1>.

Transcript Area Menu

The transcript area menu, shown in Figure 1-11, contains commands that manipulate text in the transcript area, interrupt DSEE commands, and exit the transcript area.

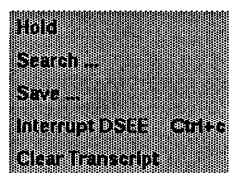


Figure 1-11. Transcript Area Menu

The transcript area menu is available in pop-up form only. You can display it by pointing the cursor anywhere within the transcript area and pressing and holding <M3>.

The transcript area menu offers the following selections:

- | | |
|-------------------------|---|
| Hold | Stops the transcript area from scrolling up when you issue commands. This enables you to view text in the transcript area while a command is in progress. To deselect Hold , select Release Hold from the transcript area menu. (When you select Hold , Release Hold appears on the menu in place of Hold .) |
| Search | Searches forward or backward in the transcript area for a text string that you specify. |
| Save | Writes the contents of the transcript area into a file that you specify. |
| Interrupt DSEE | Cancels a DSEE command that is already in progress. |
| Clear Transcript | Removes all text from the transcript area. Clear Transcript is useful when a large amount of text in the transcript area makes it difficult to scroll through the text. |

Sample DSEE Session

Earlier in this chapter you selected a working directory and installed the online portion of this manual as the subdirectory **dsee_examples**. This subdirectory contains a library (which, in turn, contains nine elements), a system directory, and the subdirectory **update** (which is for administrative purposes only).

The objects in **dsee_examples** represent the starting point for the sample DSEE session, which runs the entire length of this manual. The sample session uses these objects to demonstrate how you can develop a program in the DSEE facility using relatively few commands. To duplicate the sample session, execute all commands as described in this manual.

The sample DSEE session begins with the following scenario:

You are a software engineer who has just joined a development project managed by the DSEE facility. Your goal is to build, test, modify (if necessary), and release the program **order**. In its current form, the program accepts a string of real numbers and displays them in descending order.

Your predecessor on the project left you with some indecipherable notes, the system directory **order_sys**, and the library **sample_library** containing the elements:

- order_main.c** The main module for the program **order**
- order_sub1.c** The submodule that reads real numbers from standard input and stores them in an array
- order_sub2.c** The submodule that arranges the real numbers in the array in descending order
- order_sub3.c** The submodule that writes the real numbers in the array to standard output
- order.h** The include file shared by the main module and all three submodules

order.sml	The system model for the program order
test	An element with no apparent function
merge_text	An element with no apparent function

NOTE: The `dsee_examples` directory also contains an element called `order_hcm.sml`, which is not part of the sample DSEE session in this manual. The element `order_hcm.sml` is a system model for building the sample system on multiple platforms. In *Using DSEE*, we use `order_hcm.sml` to demonstrate heterogeneous configuration management, that is, multiple-platform software development using the DSEE software. See *Using DSEE* for details.

DSEE Documentation

As you go through the examples in this book, you may find that you want to know more about DSEE facilities. You may want to know details about how certain functions work, for example, or you may want to know more specific information about certain DSEE commands. Answers to such questions can be found in other DSEE books.

There are three books in the DSEE documentation set. (See the Preface for information on ordering DSEE documents.)

- *Getting Started with the Domain Software Engineering Environment (DSEE)* (this book)
- *Using the Domain Software Engineering Environment (DSEE)*
- *Domain Software Engineering Environment (DSEE) Reference*

This book, *Getting Started with DSEE*, is a tutorial. We designed this book for you to use once and then set aside. This is the first book of the document set that you should use.

Using DSEE is a task-oriented look at the DSEE facility. It describes DSEE software in detail and examines several engineering groups that use DSEE software. Read this book both for a conceptual overview of DSEE facilities and as an aid in using DSEE software to perform certain tasks. Consult the book as you work in the DSEE facility and wish to know how to best use the software.

DSEE Reference is the comprehensive reference manual of the DSEE command interface. It contains complete information, including all options and arguments, on all DSEE commands; the DSEE system model, configuration thread, and model thread languages; and DSEE administration. Use this book to learn how to write system models, configuration threads, and model threads, and how to administer a DSEE facility.

DSEE Reference also contains complete information about the DSEE callable interface, which enables you to execute DSEE commands and access DSEE information from within C and Pascal programs.

In addition to these books, we provide online help files for DSEE commands, system model language declarations, and conditional processing directives. The help files also provide overviews of a number of DSEE topics, such as DSEE administration and protection.

We also provide a *DSEE Release Document* which has the latest release information about installing DSEE, documentation, and bugs.

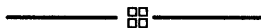
Exiting the DSEE Facility

You can use the Quit menu or the `quit` command (from the command line) to end your DSEE session at any time.

If you want to exit the DSEE facility now, point to the Quit menu title and click <M1>. From the resulting menu, select Confirm. When

you restart the DSEE facility to continue with this tutorial, your current library and working directory settings will be just as you left them.

If you want to continue with this tutorial now, turn to Chapter 2.



Prerequisites to Chapter 2

Before you perform any of the exercises in this chapter, please make sure that you have

- Installed the online examples for this manual as directed in Chapter 1.
- Invoked the DSEE environment as directed in Chapter 1.

Chapter 2

Managing Sources

As we mentioned in the previous chapter, the DSEE storage area for source code, text, and other evolving work is a **library**. Each item stored in a library is called an **element**. Elements can have several lines of descent, and each line of descent can have many versions.

The DSEE facility manages sources through tight control on libraries and elements. Every library has its own protection attributes, which identify who can and cannot access elements in the library, and how much access each user is allowed. A user who has permission to add versions to element lines of descent is prevented from attempting to add versions to the same line of descent at the same time as another user: a user reserves a line of descent, creates the new version, and then replaces the new version on the line of descent.

The DSEE facility has many commands that you can use to access libraries and elements. However, there are only a few commands that you need to know to get started with DSEE source management. These commands, which will be described in this chapter, are:

- compare**
- create branch**
- create element**
- create library**

delete element

merge

name version

read

replace

reserve

set library

show derivation

show elements

show history

We also discuss the following desktop commands and commands that allow you to invoke or emulate shell commands from the desktop:

Browse Directory

Cut Icon

Edit File

shell

cd

pwd

wd

Using Libraries

A DSEE library is a database containing elements and other objects used by the DSEE facility.

A library resembles a directory. You can list its contents and read its elements as if they were files. To create, modify, or delete an element or other DSEE object within a library, however, you must use DSEE commands.

Libraries also have protection attributes that determine who can do what with their contents. There are four classes of DSEE users: administrator, member, reader, and non-user. Generally, the person who creates a library is its “administrator,” which is the highest of the four classes. You are an administrator of the sample library that now exists in your working directory.

Setting Contexts

Earlier we said that when you invoke the DSEE facility, it restores whatever settings were current during your previous session. Not having had a previous session, you will need to establish at least one setting—your current library. This is the library that contains elements that you will be working with.

Your working directory is another important setting. Whenever the DSEE facility creates a file (for instance, when it creates an editable copy of one of the current library’s elements), it creates the file in your working directory by default. Before you set your current library, you need to set `dsee_examples` as your working directory.

You can use the UNIX `cd` command or the Aegis `wd` command to set the working directory for the DSEE desktop. Use the `cd` command to set the directory `dsee_examples` as your working directory for this sample session. (You may find it easier to issue commands such as `cd` from the desktop’s command line. You can issue from the command line any commands on the desktop menus that do not begin with a capital letter. For example, you can enter `pwd`, but not **Browse Directory** from the command line.)

1. Select Directory from the Misc. menu.
2. From the submenu that appears, select `cd` (see Figure 2-1).

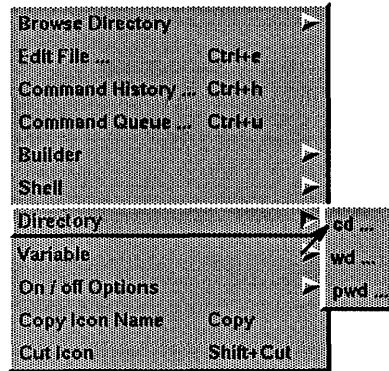


Figure 2-1. Selecting `cd`

3. The DSEE facility displays the dialog box shown in Figure 2-2.

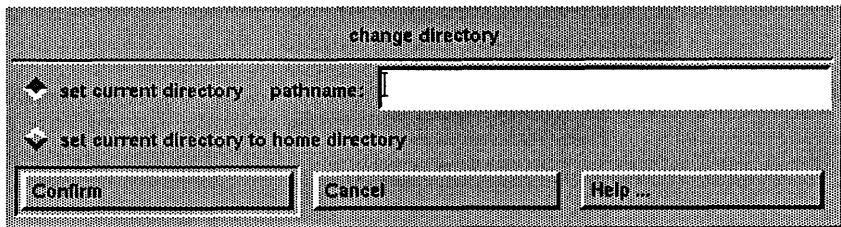


Figure 2-2. The `cd` Dialog Box

With the cursor inside the dialog box, type

pathname/dsee_examples

where *pathname* is the pathname of the directory where you installed `dsee_examples`.

4. If you make an error entering text in the dialog box, you can use <BACKSPACE> to delete the text. For a more selective edit of text, use the mouse: point just past a character you want to delete and click <M1>. Then use <BACKSPACE> to delete the character.

Point to the Confirm button and click <M1>.

After you click on the Confirm button, the DSEE facility updates the working directory banner and displays an icon representing your working directory. Figure 2-3 shows the working directory icon.

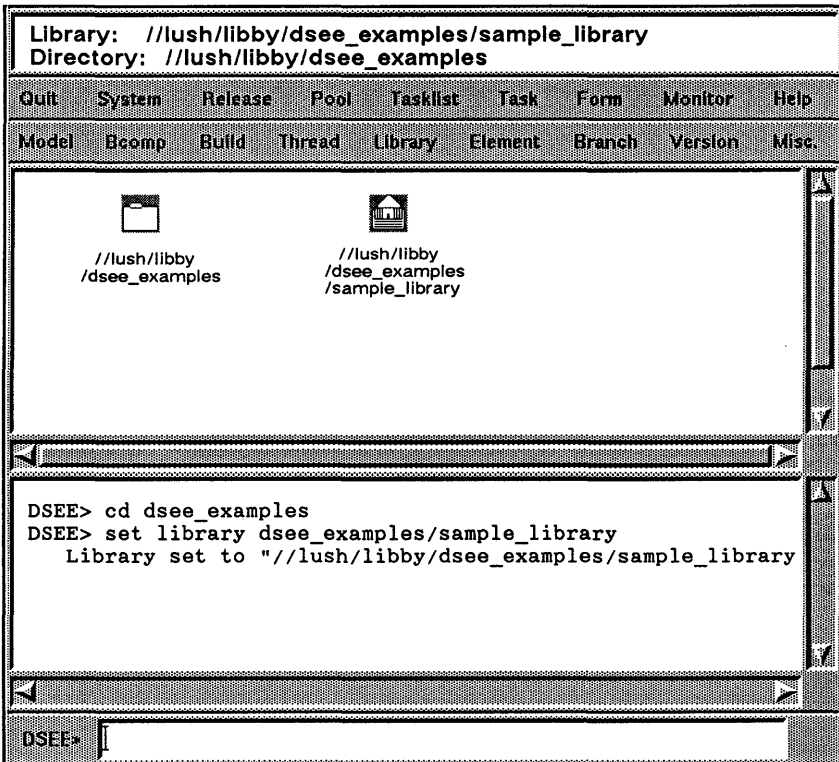


Figure 2-3. Current Library and Directory Settings

(You can also issue the UNIX `pwd` command directly from the DSEE desktop. To execute other shell commands use the DSEE shell command. See *Using DSEE* for details.)

The `set library` command sets or clears your current library setting. To set the sample library in your working directory as your current library, perform the following steps:

1. Select `set library` from the Library menu.
2. The DSEE facility displays a dialog box. With the cursor inside the dialog box, type
`sample_library`
3. Point to the Confirm button and click <M1>.

The DSEE facility displays your current library setting in the context banner. It also displays a library icon representing your current library setting as shown in Figure 2-3. In the transcript area, the DSEE prompt (“DSEE>”) precedes each command you enter.

Exploring the Contents of a Library

The Library menu’s `Browse` command lets you view a list of all the elements in the current library. To view the contents of `sample_library`,

1. Select `Browse` from the Library menu.
2. From the submenu, select `display elements`. The DSEE facility displays the `browse box` shown in Figure 2-4.

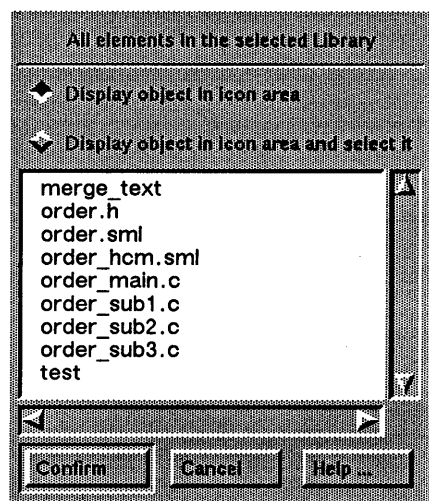


Figure 2-4. Browsing the Current Library

A **browse box** presents a list of objects that you can display as icons on the desktop. In this case, the browse box lists elements. Browse boxes can list other objects, such as reserved branches in a library or the most recent version on a branch.

You can display icons for one or more of the objects listed in a browse box. First you need to select the items you want to display. To select items in the browse box, you can do any of the following (we suggest you experiment with these methods of selection):

- To select one or more items while deselecting all previously selected items, point to an item you want to select and press and hold <M1>. Highlight any adjacent items you want to select by dragging the cursor. Release <M1> when you've highlighted the items you want to select.
- To select one or more items without deselecting previously selected items, point to an item you want to select and press and hold <CTRL> <M1>. Release <CTRL>. Highlight any adjacent items you want to select by dragging the cursor. Release <M1> when you've highlighted the items you want to select.

- To select two items and all items between the two items in the list, point to the first item and click <M1>, then point to the second item, press and hold <SHIFT>, and click <M1>.
- To deselect one item without deselecting others, point to the item, press and hold <CTRL>, and click <M1>.

Display icons for elements other than **order_hcm.sml** in **sample_library** (as we mentioned in Chapter 1, **order_hcm.sml** is not part of the sample DSEE session in this manual):

1. Select the objects listed in the **sample_library** browse box except **order_hcm.sml** using the preceding methods of selection.
2. Once you have selected the names, point to the Confirm button and click <M1>.

As shown in Figure 2-5, eight element icons appear on your desktop, representing each of the eight elements. The DSEE facility may place some of the icons out of view. Use the icon area scroll bars to view the entire contents of the icon area. Notice that there are different icons to represent directories, DSEE libraries, and DSEE elements.

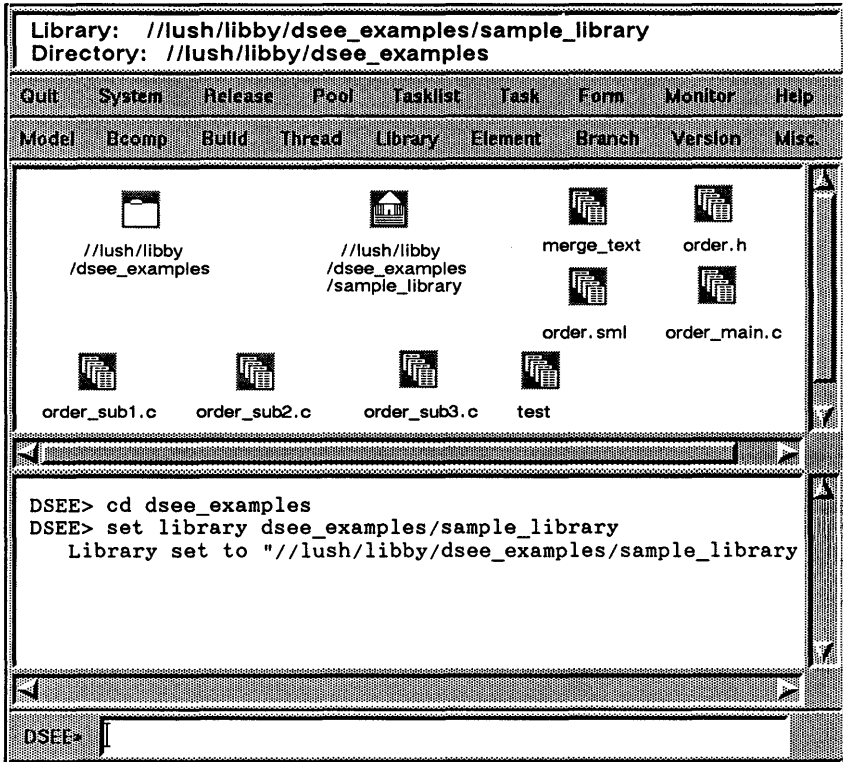


Figure 2-5. Icons on the Desktop

You can rearrange the icons on your desktop. Move the icons representing the elements `test` and `merge_text` away from the elements with names that begin with “order.”

1. Position the cursor on the `merge_text` icon, and press and hold down <M1>.
2. Move the mouse. Observe that an outline of the icon follows the mouse’s movements. When you’ve positioned the outline to your satisfaction, release <M1>. The icon moves to the new position.
3. Move the `test` icon in the same manner.

Listing the Contents of a Library

Now that you've displayed icons for elements in the current library, you can find out more about the elements using the **show elements** command. (This command, like most of the source management commands in the DSEE facility, requires a current library setting.)

When you invoke it without arguments, the **show elements** command displays (in the transcript area) the names of all elements in your current library. You can also use **show elements** as follows to display any comments entered when the elements were created.

1. If the **test** icon is selected, deselect it (point to an empty space in the icon area and click <M1>).
2. Select **show elements** from the Element menu (see Figure 2-6).

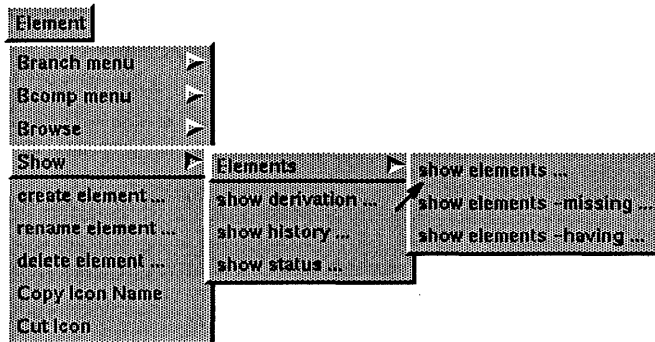
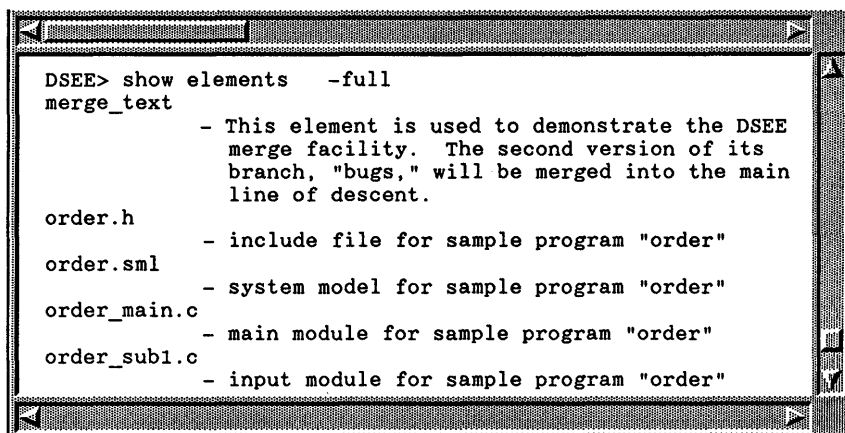


Figure 2-6. Selecting show elements

3. When the dialog box appears, click on the small button next to the **-full** option. The **-full** option displays any comments entered at the time the elements were created. To show all the elements in a library, leave the text entry box empty.
4. Point to the **Confirm** button and click <M1>.

Figure 2-7 shows part of the list of elements in `sample_library` and each element's associated comments. To view the entire list, use the transcript area's scroll bar.



```
DSEE> show elements -full
merge_text
    - This element is used to demonstrate the DSEE
      merge facility. The second version of its
      branch, "bugs," will be merged into the main
      line of descent.

order.h
    - include file for sample program "order"

order.sml
    - system model for sample program "order"

order_main.c
    - main module for sample program "order"

order_sub1.c
    - input module for sample program "order"
```

Figure 2-7. show elements -full Command

Creating a Library

The `create library` command creates a library and sets it as your current library. If you specify a leafname (that is, a name with no directory path) rather than a complete pathname for a new library, the DSEE facility creates it in your working directory.

Create the library `my_library` in your working directory:

1. Select `create library` from the Library menu.
2. The DSEE facility displays a dialog box. With the cursor inside the dialog box, type
`my_library`
3. Point to the Confirm button and click <M1>.

4. The DSEE facility displays another dialog box, which asks, "What is the function of this Library?" Since you're going to use this library later in Chapter 4 to hold an element, you might want to enter the comment "Creating second library for system." Your comment is recorded in the DSEE facility's historical information about the library.
5. When you're satisfied with your comment, click the Confirm button.

In the transcript area, messages indicate that the new library has been created and that Domain/OS protections have been set up for it. In the context banner at the top of the DSEE window, the legend indicates that the new library is the current library. In the icon area, a new library icon appears, representing the new library.

Using Elements

An element is a text file stored inside a DSEE library. Every time you modify an element, the DSEE facility creates a new version of the element, assigns it a version number, and stores it in compressed form along with all previous versions of the element. You can recall any element version by specifying the version number with the element name.

To examine the elements provided with this manual, set your current library context back to **sample_library**:

1. Point to the icon for **sample_library** and click <M1>.
2. With the cursor in the icon area, display the pop-up menu by pressing and holding <M3> or by clicking <M3>. In Chapter 1 we mentioned that, when no icons are selected, the pop-up menu is the same as the Misc. pull-down menu. Now that you've selected a library icon, the pop-up menu will be the same as the Library pull-down menu.
3. Select **set library** from the Library menu.
4. When the dialog box appears, the path to **sample_library** is in the text entry box. Point to the Confirm button and click <M1>.

Examining Elements

The **read** command can display a read-only copy of any element version stored in your current library. For example, to read the latest version of the element **test**, perform the following steps:

1. Point to the icon for **test** and click <M1>.
2. Select the **read** command from the Version menu.
3. The DSEE facility displays a dialog box seeded with the name “test.” Point to the Confirm button and click <M1>.

The DSEE facility displays a read-only copy of **test** using the editor specified by the X resource **Dsee*read**. (The file `/usr/X11/lib/app-defaults/Dsee` lists the X resources for the DSEE software. See *DSEE Reference* for details.) If you have not specified an editor, the DSEE facility invokes **xedit** by default.

4. Once you have finished viewing **test**, exit the editor.

To read an earlier version of **test**, you need to specify a version number. For example, to read version 1 of **test**, perform the following steps:

1. Select the **read** command from the Version menu.
2. When the dialog box appears, it is seeded with the name “test” because the **test** icon is still selected. Position the cursor immediately after the end of the name “test” and click <M1>. This enables you to append text to the text already present in the dialog box.
3. Type
[1]
4. Point to the Confirm button and click <M1>.
5. Once you have finished viewing **test[1]**, exit the editor.

Examining Element Histories

Every time someone creates or modifies an element, the DSEE facility requests a comment on the operation. These comments, along with information supplied by the DSEE facility such as the date and time of the operation, constitute an element's history.

You can examine the histories of elements in your current library with the **show history** command. For example, examine the history of **order.sml**:

1. Point to the icon for **order.sml** and click <M1>.
2. With the cursor in the icon area, display the pop-up menu by pressing and holding <M3>.
3. The DSEE facility displays the Element menu. Select **show history** and release <M3> (Figure 2-8).

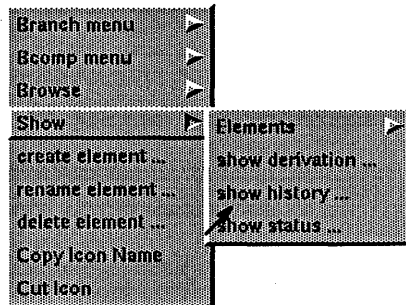


Figure 2-8. Selecting the show history Command

4. The DSEE facility displays a dialog box seeded with the name "order.sml." Point to the Confirm button and click <M1>.

The DSEE facility displays the history of the **order.sml** element in the transcript area.

Using wildcards, you can issue one command to examine the histories of several elements. Examine the histories of **order's** C source files:

1. Display the pop-up menu by pressing and holding <M3>.
2. Select the **show history** command and release <M3> (Figure 2-8).
3. In the dialog box, point just past **order.sml**, click <M1>, and use <BACKSPACE> to erase the element name. Then type the following element name:

```
order?*.c
```

This element name uses the wildcard combination **?***, meaning "any characters."

4. This time, display detailed histories by issuing the **show history** command with the **-full** option. Click on the button to the left of the **-full** option.
5. Point to the Confirm button and click <M1>.

The DSEE facility displays the histories of all of the C source modules in chronological order. The list of comments is longer than the length of the transcript area, so some of the text scrolls off the top of the transcript area. Use the scroll bars to view the text.

Modifying Elements

You cannot directly edit an element. Instead, you must edit a copy of the element and then convert the edited copy into a new version of the element.

To modify an element, you must first use the **reserve** command. This command creates a copy of the element's latest version in your working directory. Once you've reserved an element, no one else can reserve it until you replace the element or cancel the reservation.

Once the DSEE facility creates a copy of an element in your working directory, you can edit the copy just as you would edit any other file.

Next, you use the **replace** command. The **replace** command copies the contents of the edited file to the reserved element, deletes the file, and releases the element for another reservation.

Each time you reserve and replace an element, you create a new version of the element. This new version becomes the latest version in a series of versions that make up the main line of descent for that element. None of the previous versions of that element are lost (unless you explicitly delete them).

(As we discuss later in this chapter, you can also create alternate lines of descent called **branches**.)

NOTE: The DSEE desktop sometimes uses the term “branch” as a shorthand for “line of descent,” which can mean either the main line or alternate line of descent. For instance, the desktop may prompt you to enter a *branch_path*, in which case you would have the choice of entering a main line or alternate line of descent. Technically speaking, however, the main line of descent is not a branch.

Create a new version of the element test:

1. Point to the icon for **test** and click <M1>.
2. Select **reserve** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with “test,” the name of the **test** element’s main line of descent. Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box; this one prompts you to enter a comment about the reservation. Type an appropriate comment, such as

Learning about reservations

5. Point to the Confirm button and click <M1>.

The DSEE facility places a branch icon, shown in Figure 2-9, in the icon area. (Recall that you may have to scroll the icon area to reveal

new icons.) The branch icon represents the reserved line of descent of the element `test`.



`test`

Figure 2-9. The Branch Icon for test

You now have an editable copy of the element's latest version in your working directory. The name of the element and its corresponding editable file are the same by default. To edit the copy, first display an icon for the file:

1. Point to the directory icon for `dsee_examples` and click <M1>.
2. Select **Browse Directory** from the Misc. menu. From the submenu that appears, select **display files**.
3. The DSEE facility displays a browse box containing the pathnames of all the files in `dsee_examples`. (The `test` file may be the only file in `dsee_examples`.) Point to the pathname for `test` and click <M1>.
4. Point to the Confirm button and click <M1>.

The DSEE facility displays a file icon for `test`. The file icon is shown in Figure 2-10.



```
//lush/libby  
/dsee_examples  
/test
```

Figure 2-10. The File Icon for test

Now invoke an editor and edit the copy of **test**:

1. Point to the file icon for **test** and click <M1>.
2. Select Edit File from the Misc. menu.
3. The DSEE facility displays a dialog box seeded with the pathname for **test**. Point to the Confirm button and click <M1>.

The DSEE facility displays **test** using the editor the X resource **Dsee*editor** specifies. (The file **/usr/X11/lib/app-defaults/Dsee** lists the X resources for the DSEE software. If you have not specified an editor, the DSEE facility invokes **xedit** by default. See the *DSEE Reference* for details.)

4. Because the actual contents of **test** have no significance in this exercise, just add some text, then save and close the file.

Now replace the updated file as the new version of the element **test**:

1. Point to the branch icon for **test** and click <M1>.
2. With the cursor in the icon area, display the pop-up menu by pressing and holding <M3>.
3. The DSEE facility displays the Branch menu, since the icon you selected is a branch icon. Select **replace** and release <M3>.
4. The DSEE facility displays a dialog box seeded with “test.” Point to the Confirm button and click <M1>.
5. The DSEE facility displays another dialog box; this one prompts you to enter a comment about your modifications to the element. The dialog box is seeded with the comment you entered when you reserved the element. Edit the comment if you wish; when it’s satisfactory, point to the Confirm button and click <M1>.

The DSEE facility replaces the main line of descent for **test** and deletes the file **test** from your working directory. Your edited copy of **test** becomes the main line’s latest version. The icon for **test**,

however, remains in the icon area. (Icons remain in the icon area unless you explicitly remove them.)

To compare the differences between the two versions, use the **compare** command. By default, this command shows differing lines side by side; if the lines do not fit in the transcript area, they are truncated. To avoid this problem, you can either make your window wider before executing the **compare** command, or you can use the **-serial** option to the command. This option ensures that you will see all of every differing line (rather than just a partial line) in the transcript area.

1. Select the branch icon for **test** if it isn't already selected.
2. Select **compare** from the Version menu.
3. The DSEE facility displays the dialog box shown in Figure 2-11. The dialog box contains three windows for entering text. The top window is seeded with "test" (because the **test** icon is still selected). Position the cursor immediately after the end of the name "test" and click <M1>. Type

[2]

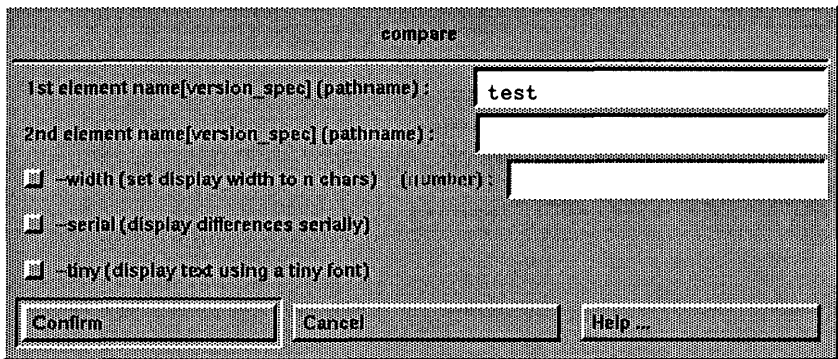
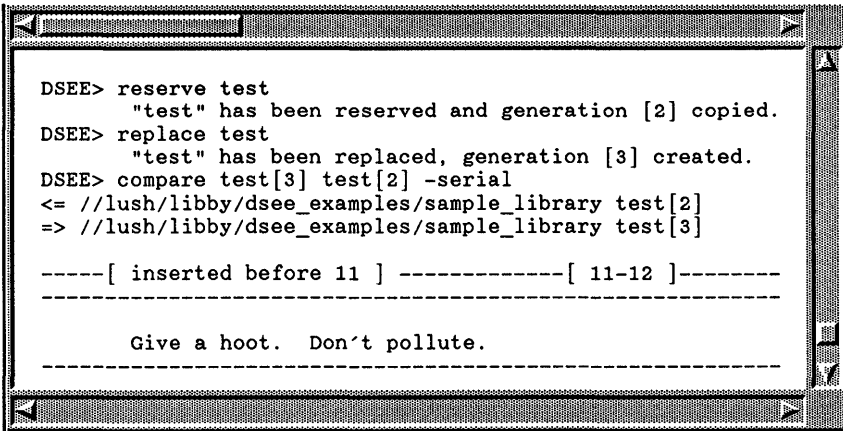


Figure 2-11. The **compare** Dialog Box

4. Point to the second empty window just below the top window, click <M1>, and type
`test[3]`
5. Click on the button next to the `-serial` option.
6. Point to the Confirm button and click <M1>.

Figure 2-12 shows the messages resulting from the `reserve` and `replace` commands, plus the `compare` command display.



```
DSEE> reserve test
      "test" has been reserved and generation [2] copied.
DSEE> replace test
      "test" has been replaced, generation [3] created.
DSEE> compare test[3] test[2] -serial
<= //lush/libby/dsee_examples/sample_library test[2]
=> //lush/libby/dsee_examples/sample_library test[3]

-----[ inserted before 11 ] -----[ 11-12 ]-----
-----
          Give a hoot.  Don't pollute.
-----
```

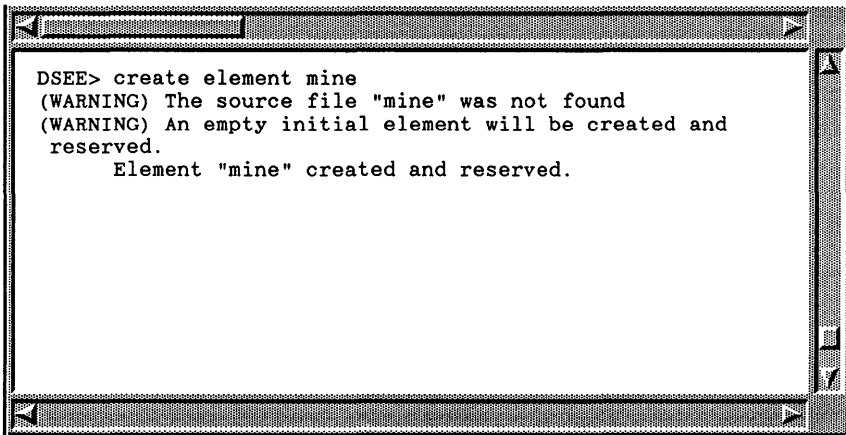
Figure 2-12. reserve, replace, and compare Commands

Creating Elements

When you create an element with the `create element` command, the DSEE facility looks in your working directory for a file with the same name as the element. If the file exists, the DSEE facility copies the file's contents to the initial version of the new element in your current library and deletes the file. Otherwise, the DSEE facility creates and reserves an empty element without creating a corresponding file in your working directory.

To create and reserve the empty element **mine** in your current library, perform the following steps:

1. If the **test** icon is still selected, deselect it (point to an empty space in the icon area and click <M1>).
2. Select **create element** from the Element menu.
3. When the dialog box appears, type
mine
and click on the Confirm button.
4. As with most other DSEE commands that create objects, the DSEE facility displays another dialog box and requests a comment on the purpose of the element. (Had a file named **mine** existed in your working directory when you issued the command, the DSEE facility would have displayed a second dialog box for a comment on the first version of the element.) Enter your comment.
5. Click on the Confirm button to continue creating the element. The DSEE facility displays a branch icon for **mine** in the icon area. Figure 2-13 shows the output in the transcript area.

A screenshot of a terminal window with a standard Mac OS-style title bar and window controls. The text inside the window shows the execution of the 'create element mine' command. It includes two warning messages: one stating that the source file 'mine' was not found, and another stating that an empty initial element will be created and reserved. The final output line indicates that the element 'mine' has been successfully created and reserved.

```
DSEE> create element mine
(WARNING) The source file "mine" was not found
(WARNING) An empty initial element will be created and
reserved.
      Element "mine" created and reserved.
```

Figure 2-13. create element Command

Creating Branches

Repeated reserve and replace operations on an element create a line of descent. An element's line of descent consists of the latest version of that element and all previous versions.

An element can also have alternate lines of descent called **branches**. Branches typically develop because someone has reserved the element's main line of descent or because you need to make changes to an older version of the element. Like the element's main line of descent, a branch may also consist of multiple versions created by successive reserve and replace operations.

When you create a branch with the **create branch** command, you specify the branch name and the element version from which the branch originates. The DSEE facility creates the branch, reserves it in your name, and copies the specified element version to an editable file in your working directory. Note that the editable file has the same name as the element and not the branch.

As an example of why you might create a branch, suppose that the element **test** in your sample library is one of the source files of a software product. The current release of the product uses **test[2]**. You have already modified the element for the next version of the product, thus creating **test[3]**.

Someone in the field reports a serious bug in **test[2]**. Because your modifications in **test[3]** represent a significant shift in design, you cannot fix the bug by creating a new version in the element's main line of descent. Instead, you must modify the version currently used in the field.

You therefore use the **create branch** command to create a branch called **bugfix** growing from the element version **test[2]**.

1. Point to the branch icon for **test** and click <M1>.
2. Select **create branch** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with the name "test." Position the cursor immediately after the end of the name "test" and click <M1>. This enables you to

append text to the text already present in the dialog box.
Type

[2]

4. Point to the empty text entry box next to “branch name:” and click <M1>. Now type

bugfix

5. Point to the Confirm button and click <M1>.
6. The DSEE facility displays a dialog box requesting a comment on the purpose of the branch. Type an appropriate comment, such as, “Fix bug 345.” and click on the Confirm button. A branch icon representing **test/bugfix** appears in the icon area.

The DSEE facility creates an empty branch and copies the specified element version to a file named **test** in your working directory. You make the necessary changes to the file (edit and close the file) and issue the **replace** command for the branch.

1. Point to the branch icon for **test/bugfix** and click <M1>.
2. Select **replace** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with the name “test/bugfix.” Click on the button next to the **-nc** option. The **-nc** (no comment) option tells the DSEE facility that you don’t want to comment on the operation.
4. Point to the Confirm button and click <M1>.

The DSEE facility makes your modified file **test** the initial version of the branch **test/bugfix** and issues a message confirming the creation of the new branch’s first version. You then recompile the released product using **test/bugfix[1]** and send the results out to the field.

Figure 2–14 illustrates the current state of the element **test**. Note that, except at creation time, you refer to a branch in DSEE commands with the syntax *element_name/branch_name* (for example, **test/bugfix**).

To see how the DSEE facility has recorded information about the branch in the library's history database, issue the **show history** command:

1. Point to the element icon for **test** and click <M1>.
2. Select **Show** from the Element menu.
3. From the submenu that appears, select **show history**.
4. The DSEE facility displays a dialog box seeded with "test." Since you selected the **-full** option the last time you issued the **show history** command, **-full** is still selected (unless you exited and restarted the DSEE facility). Deselect the **-full** option.
5. Point to the Confirm button and click <M1>.

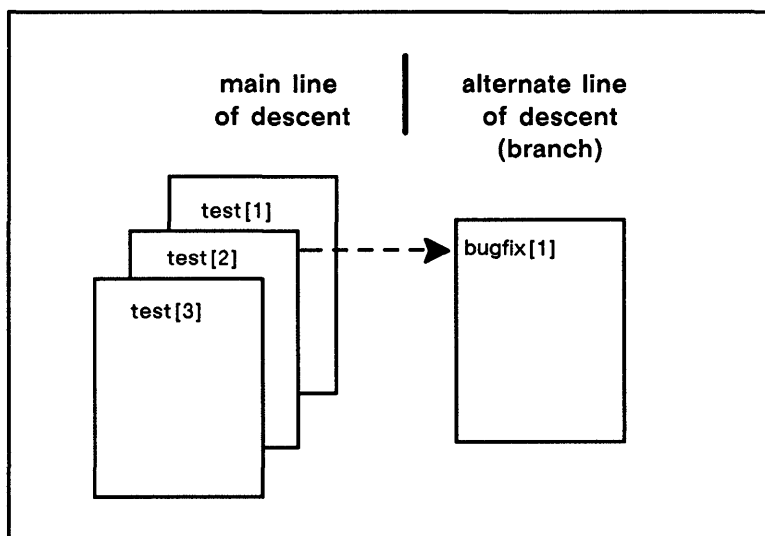


Figure 2-14. Representation of Element test with Branch

Merging Branches

If you're working on both a branch line of descent and the main line of descent, eventually you may want to bring all of the work together. For example, in the situation described above, in which you made bug fixes on a branch, you'd probably want to make sure that those bug fixes were incorporated in the work done on the main line of descent.

The DSEE **merge** command gives you the ability to merge lines of descent. It incorporates the contents of a version on one line of descent (called the **source version**) into another line of descent of the same element (called the **target line of descent**). The result is a new version on the target line of descent.

The **merge** command tries to build the new version of the element for you automatically. If it can't automatically decide what to do at any point in the merger, the DSEE facility asks you to intervene. During the merger, and after the merger is complete, the DSEE facility reserves the new version and displays its contents in an edit window. Once you've reviewed the text, you can replace it in the main line of descent with the **replace** command.

The merger algorithm works as follows: the DSEE facility compares the source version and the most recent version on the target line of descent to the youngest ancestor of the two (called the **base version**). If you were merging the bug fixes in **test/bugfix[1]** in the last section into the main line of descent of the **test**, your base version would be **test[2]**, since it's the youngest ancestor of both the source version (**test/bugfix[1]**) and the target line of descent's most recent version (**test[3]**). (Figure 2-15 depicts a base version, a source version, and a target line of descent on an element.)

The DSEE facility builds the merged version by reading the base version and the other two versions from top to bottom. Where the three versions are the same, the merged version contains that text. If either the source version or the most recent version on the target line of descent contains a change or addition to a part of the text of the base version, the DSEE facility puts the changed or added text into the merged version. If both the source version and the most recent version on the target line of descent differ from the base version, but they also differ from one another, the DSEE facility asks you which of the changed lines you want in the merged version.

In order to give you some practice with the merger process, we've placed an element named `merge_text` in the sample library. It has a branch named `bugs`. To practice a merger, you will merge `merge_text/bugs[2]` into the main line of descent. Figure 2-15 shows a representation of the current state of `merge_text` and the state of `merge_text` after the merger.

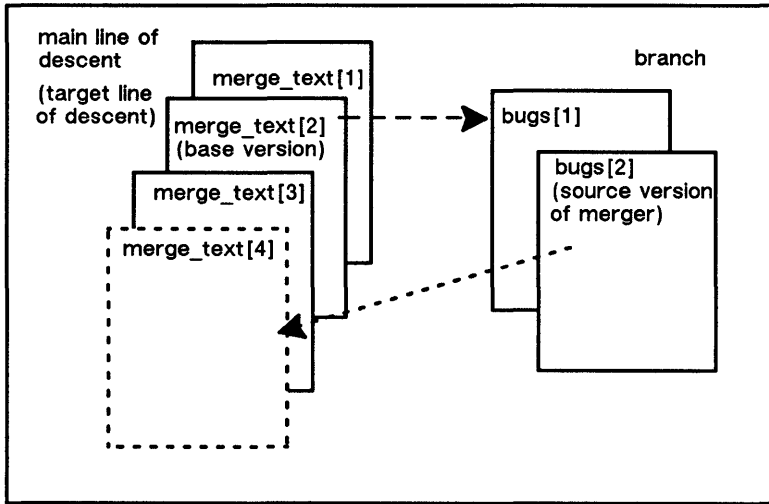


Figure 2-15. Representation of a Merger

Before you perform the merger, you need to place an icon for the `merge_text/bugs` branch in the icon area. You can list branches with the name `bugs` using the **Browse** command.

1. If the **test** icon is still selected, deselect it.
2. Select **Browse** from the Library menu.
3. From the submenu that appears, select **display branch paths with a particular branch name**.
4. When the dialog box appears, type

`.../bugs`

indicating any branch path that ends with "bugs."

5. Point to the Confirm button and click <M1>.
6. When the browse box appears, select `merge_text/bugs`.
7. Point to the confirm button and click <M1>. The DSEE facility places a branch icon for `merge_text/bugs` in the icon area.

Now you can merge `merge_text/bugs` into the main line of descent.

1. Select the icon for `merge_text/bugs`.
2. Select **Merge** from the Version menu.
3. From the submenu that appears, select **merge -reserve**. The keyword **-reserve** tells the DSEE facility to put the merged text into a reserved version of the element.
4. The DSEE facility displays a dialog box seeded with the latest version of the `bugs` branch and the main line of descent (`merge_text`).
5. Point to the Options button and click <M1>. This displays all the options available.
6. Click on the button next to the **-serial** option. The **-serial** option tells the DSEE facility to show differing lines one after the other rather than side by side.
7. Point to the Confirm button and click <M1>.
8. The DSEE facility displays a dialog box that is seeded with some information about the merge and that asks why you are merging the versions. Type a comment about the merger, then click on the Confirm button.

NOTE: When the DSEE facility can make automatic decisions, the merger will proceed very quickly. The process is suspended only when the DSEE facility can't make an automatic decision. To find out about decisions that have been made automatically, scroll up through the transcript area.

When you click on the Confirm button, the merger process creates an edit window into which the merged text will be written. It is the transcript area, however, that displays the changed lines in `merge_text[3]` and `merge_text/bugs[2]` and the automatic decisions that the DSEE facility makes about these lines.

If the DSEE facility cannot automatically determine which version's text to put into the merged version, it will tell you in the transcript area why it can't decide and display a query box asking you which lines it should use. Figure 2-16 shows what happens the first time that the DSEE facility can't make an automatic decision in this example.

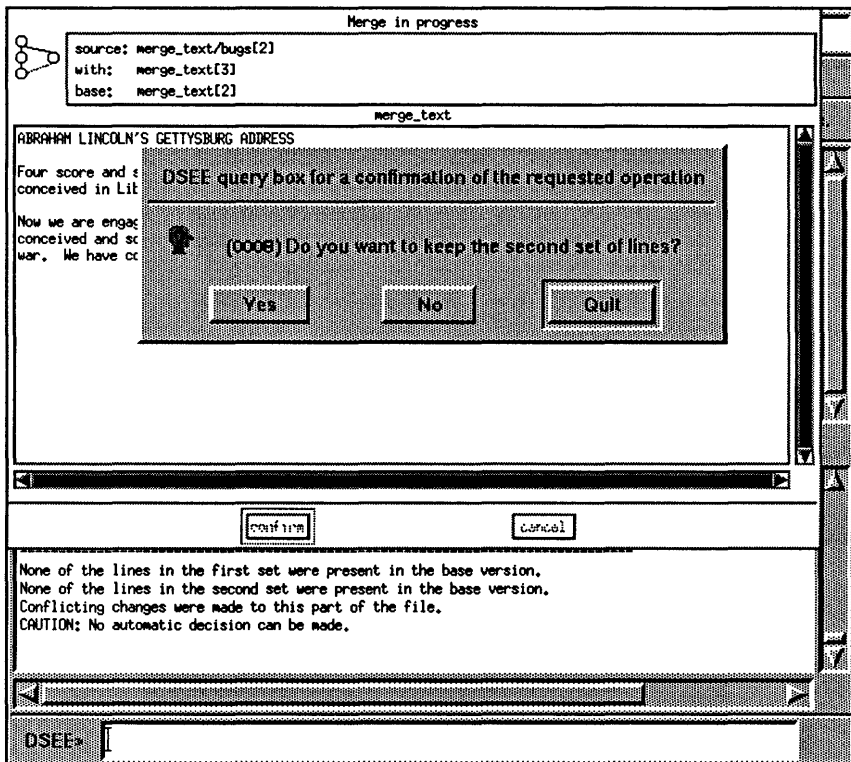


Figure 2-16. Query Box for merge Command

9. The query box asks, “(0008) Do you want to keep the second set of lines?” (The “0008” indicates that the merger stopped at the eighth line of text.) Point to the Yes button and click <M1>.

NOTE: The edit window may obscure the differing lines of text displayed in the transcript area. Move the edit window to view the text.

10. The DSEE facility displays another query box; this one asks, “(0009) Do you want to keep the first set of lines?” Point to the No button and click <M1>.

After you’ve answered, the DSEE facility tells you (in the transcript area) that you might want to edit the file directly, and it suspends the merger process so that you can make any changes. This gives you an opportunity to make adjustments to the merged text right away. (The ability to edit on the spot is very useful and important. In many instances in which both texts in the merger differ from the base version, you’ll need to edit the final text to ensure accuracy. For example, if you’re merging differing versions of source code, the text that gets constructed during a merger might contain compilation errors.)

The DSEE facility also displays the dialog box shown in Figure 2-17. You will use this dialog box later to restart the merger process.

11. The edit window contains all of the text of the merged version that the DSEE facility has constructed to this point. You’ll see that “It is very fitting and proper” is the last line of the text.

To change the word “very” to “altogether,” position the cursor after the word “very” and click <M1>. Use <BACKSPACE> to remove the word “very,” then type

altogether

12. To restart the merger process, point to the Dismiss button and click <M1> (see Figure 2-17).

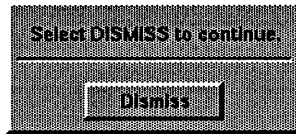


Figure 2-17. Restarting the Merger Process

13. This example contains two instances in which the DSEE facility can't automatically decide what to put in the merged text. You've just completed the first. In the second situation, tell the DSEE facility to use the lines that contain the word "people" instead of the lines that contain the word "citizens." You won't need to edit this line afterwards; simply restart the merger process as you did in step 12.
14. Once the DSEE facility has merged the text, you can still make further edits. Move the cursor to the edit window and make changes you wish. When you're done, point to the edit window's Confirm button and click <M1>.

Now that you've closed the edit window, you need to replace the new version of `merge_text` in the library.

1. Point to the branch icon for `merge_text` and click <M1>.
2. Select **replace** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with "merge_text." Since you selected the `-nc` option when you last issued the **replace** command, `-nc` should still be selected. If not, select it now.
4. Point to the Confirm button and click <M1>.

Once you're done with the merger, you may want to see a graphic representation of the changes to the element.

1. Select **show derivation** from the Branch menu.
2. When the dialog box appears, it is seeded with "merge_text" since the **merge_text** icon is still selected. Point to the Confirm button and click <M1>.
3. The DSEE facility opens the window shown in Figure 2-18. The window displays a graphic representation of changes to the element. When you finish viewing the picture, point to the Dismiss button and click <M1>.

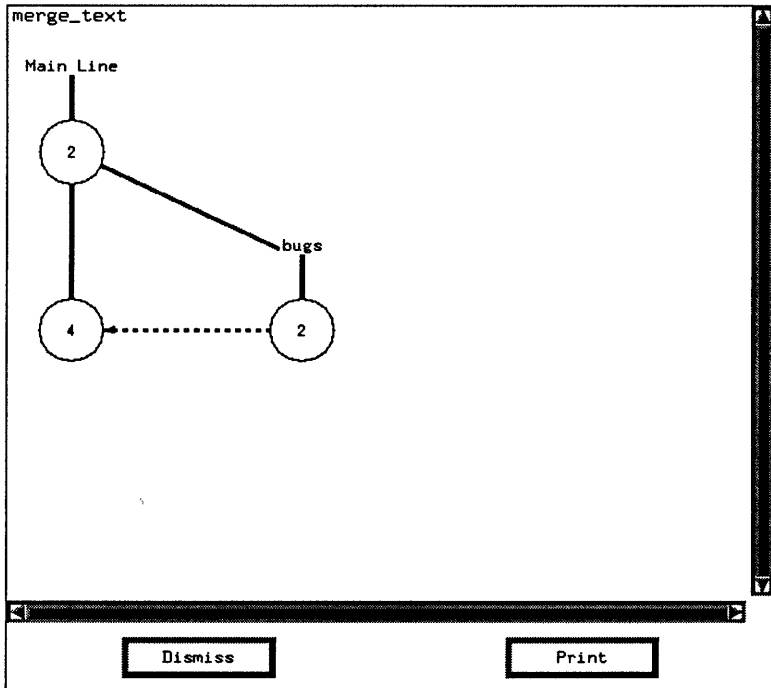


Figure 2-18. show derivation Window

The **merge** command enables you to combine versions on different lines of descent in many ways. Any version on a line of descent can be merged into another line of descent—you don't have to merge in the most recent version, as we've done here. Also, you can perform "incremental" mergers, incorporating changes performed on one line of descent into another at several stages in the development process. This technique minimizes the number of differences between the source version and the most recent version on the target line of descent at the final merger. As a result, you will have less work to do to make the merged versions compatible.

Naming Element Versions

So far, you've referred to element versions by name and number, for example, **test[1]**. Because the elements in your sample library consist of only one version each, remembering which element versions make up the program **order** is a simple matter. However, as you develop new versions of your program, associating the correct versions of elements with those programs becomes more complex.

For this reason, among others, the DSEE facility supports version names. If you assign one version name to a group of related element versions, for example, you can refer to those element versions later using their common version name. (Note that you cannot assign one version name to two versions of the same element; however, you can assign different version names to the same element version.)

The **name version** command assigns a version name to one or more element versions. Use it to assign the version name **sample** to the latest version (indicated by "[]") on the main line of descent of all elements in your current library.

1. If the **merge_text** icon is still selected, deselect it (point to an empty space in the icon area and click <M1>).
2. Select **name version** from the Version menu.
3. The DSEE facility displays a dialog box. In the top text entry box, type

?*[]

4. Still in the dialog box, point to the text entry box next to the words “version name:” and click <M1>. Type

sample

5. Point to the Confirm button and click <M1>.
6. The DSEE facility displays the query box shown in Figure 2–19. The query box requests confirmation for the elements matching the wildcard specification. You can either click on the Yes button to assign the version name to one element at a time, or you can click on the Go button to assign the version name to all the elements at once.

Point to the Go button and click <M1>. The DSEE facility assigns the version name **sample** to all the elements in **sample_library** but one. (The element **mine** has no version—because you created it and never replaced it—and is therefore ignored.)

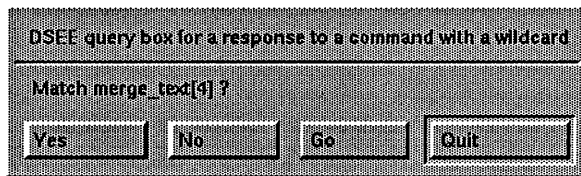


Figure 2–19. name version Query Box

You can use the **show version** command to display all element versions associated with the version name **sample**.

1. Select **show version** from the Version menu.
2. In the top text entry box, type

?*[sample]

(Note that there is no blank space between the asterisk and the left square bracket.)

3. Point to the Confirm button and click <M1>.

This command, besides displaying names, provides additional information on each element version involved.

Deleting Elements

To delete the practice elements **mine** and **test**, use the **delete element** command.

1. Select the element icon for **test**.
2. Select **delete element** from the Element menu.
3. When the dialog box appears, it is seeded with “test” since the **test** icon is still selected. Point to the Confirm button and click <M1>.
4. The DSEE facility displays the element’s history in the transcript area. The DSEE facility also displays a query box that asks if you’re sure that you want to delete **test**. Point to the Yes button and click <M1>.
5. Repeat the procedure for the elements **mine** and **merge_text**.

Removing Icons

Even though you deleted the elements **test**, **mine**, and **merge_text**, the icons for those elements are still displayed in the icon area. To remove the icons, use the **Cut Icon** command.

1. Point to the icon for **mine** and click <M1>.
2. To select multiple icons, use <SHIFT>. Press and hold <SHIFT>, and click on each of the icons for **test** and **merge_text** (including **test/bugfix** and **merge_text/bugs**).
3. Select **Cut Icon** from the Branch menu.

Command Summary

All of the commands described in this chapter have one or more options. To learn about the options available with a particular command, consult the *DSEE Reference* or use the **DSEE help** command.

Here is a brief list of the commands that you've used in this chapter and their definitions. We group the commands according to the menus from which you can invoke them.

Branch

- create branch** Creates an alternate line of descent for an element.
- Cut Icon** Removes selected icons from the desktop icon area. (Desktop menu only.)
- replace** Creates a new version of an element from an edited copy.
- reserve** Reserves a line of descent and creates an editable copy of the latest version.
- show derivation** Graphically shows an element's lines of descent, mergers, and named versions.
- show history** Shows the histories of one or more elements.

Element

- create element** Creates a new element in the current library.
- delete element** Deletes an element and its history from the current library.

Element

Show

Elements

- show elements** Displays one or more elements.

Library

- Browse** Displays icons on the desktop that represent various types of objects. (Desktop menu only.)
- create library** Creates a DSEE library.
- set library** Sets or clears the current library setting.

Misc

- Edit File** Displays a file for editing. Edit File invokes the editor that you specify with the X resource `Dsee*editor`. The default editor is `xedit`.

Misc

Directory

- cd** Sets the current working directory.
- pwd** Displays the current working directory.
- wd** Sets or displays the current working directory.

Misc

Shell

- shell** Enters a shell interactively or executes shell commands from the DSEE facility.

Version

- compare** Displays the differences between two versions, two files, or a version and a file.
- name version** Assigns a name to a particular version of one or more elements, or deletes a previously assigned version name.
- read** Displays a read-only copy of the specified element or branch version using the editor that you specify with the X resource `Dsee*read`. The default editor is `xedit`.

Version

Merge



- merge** Merges element versions, branch versions, or files.

Related Information

There are many other DSEE commands pertaining to the use of libraries and elements than we've presented here. In general, you would use these commands with less frequency than the ones described in this chapter. We group these commands according to the menus from which you can invoke them.

Branch

More branch cmds



- cancel obsolete** Activates a line of descent that was previously deactivated.
- cancel reserve** Cancels a reservation of a line of descent.
- delete branch** Deletes a branch and its sub-branches.

obsolete Declares a line of descent obsolete.

rename branch
Renames a branch.

Build Environment

set environment
Sets or clears the source reference environment for one or more elements.

show environment
Displays the current source reference environment for one or more elements.

Element

rename element
Renames an element.

Element Show

show status Shows the status of one or more elements.

Library Administer

delete library Deletes the current library.

protect library
Sets or shows the current library's protection attributes.

recover library
Brings the current library's database and history files into a consistent state after a crash or network failure.

recover user Cleans up incomplete library operations of the specified user.

reformat library

Converts a library database to suit new DSEE software.

share Allows other users to access the current library, or prevents them from accessing the current library.

show users Displays information about the users accessing the current library.



show branches

Displays the branch leafnames used by one or more elements.

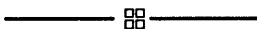
show reservations

Displays reserved elements in the current library.



delete version Deletes versions on a line of descent.

fetch Retrieves a read-only copy of an element version.



Prerequisites to Chapter 3

Before you perform any of the exercises in this chapter, please make sure that you have

- Set your current working directory to the `dsee_examples` subdirectory as directed in Chapter 2.
- Used the DSEE `set library` command to set your current library to `sample_library` as directed in Chapter 2.

Chapter 3

Building a Program

Given a system directory, a system model, and a configuration thread, the DSEE facility can automatically build an entire program or a single component upon request.

During a build, the DSEE facility extracts information from both the system model and the configuration thread to locate sources and produce derived objects. The DSEE facility also creates a detailed build map or **Bound Configuration Thread (BCT)** for each derived object it builds. The DSEE facility then stores the derived objects and their corresponding BCTs in a **binary pool** for use during rebuilds.

The commands discussed in this chapter are:

build

edit thread

examine build

examine thread

export

set model

set system
set thread
show builds

Preparing for a Build

Before you can build all or part of a program in the DSEE facility, your working context must include a current system, a current system model, and a current configuration thread (all of which will be explained in this chapter). The DSEE facility requires you to set the system first, then the system model, and finally the configuration thread.

The System

In general, a **system** is a collection of software modules combined to produce one or more executable programs. The sample program **order** is a system by this definition.

In the DSEE facility, the term “system” also refers to a directory. The DSEE facility uses a system (directory) to store the internal information and objects needed during a build. Every program you build in the DSEE facility must have its own system.

The **set system** command sets an existing system as your current system. The command also restores any working context settings (for example, your current system model and current library) that were in effect the last time you used the system.

The installation procedure in Chapter 1 created the system **order_sys** in your working directory. To set **order_sys** as your current system, perform the following steps:

1. Select **set system** from the System menu.
2. The DSEE facility displays a dialog box. With the cursor in the dialog box, type
`order_sys`
3. Point to the Confirm button and click <M1>.

As Figure 3-1 indicates, the DSEE facility identifies your current system in the context banner.

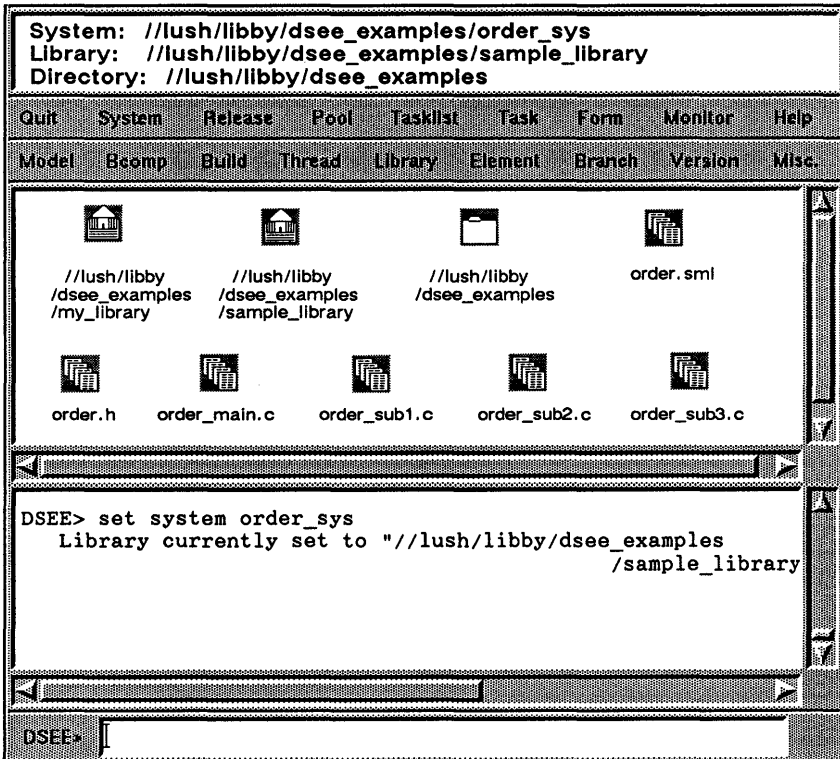


Figure 3-1. set system Command

The System Model

Once you've established a current system, you then set your current system model.

A system model tells the DSEE facility how to build a particular program. It identifies the sources, tools, and building procedures required to generate an executable version of the program. More precisely, a system model describes:

- The components of a program (the source, include, and executable files used in building the program)
- The dependencies of each component (the files and subcomponents used in building the component)
- The rules for translating the components into executable files (for example, compiler and binder commands or shell command files)

The DSEE facility provides a special language for writing system models. Like many programming languages, the system model language is symbolic and block-structured. Figure 3-2 and Figure 3-3 illustrate the block structure of a system model.

NOTE: Project programmers should have a basic understanding of the structure and function of a system model in case they need to add or change a dependency in the model. The only person who normally needs to know more about system models is the one who writes the model.

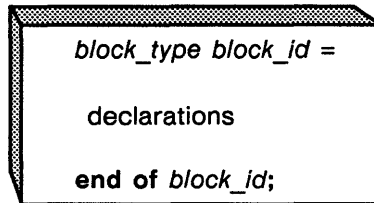


Figure 3-2. A System Model Block

In general, a **block** is a group of declarations that describe a buildable component of the program. A block begins with a keyword identifying the block type followed by the **block ID** and an equal sign. A block ends with the keyword **end** followed by a semicolon, or the keywords **end of** followed by the block ID and a semicolon.

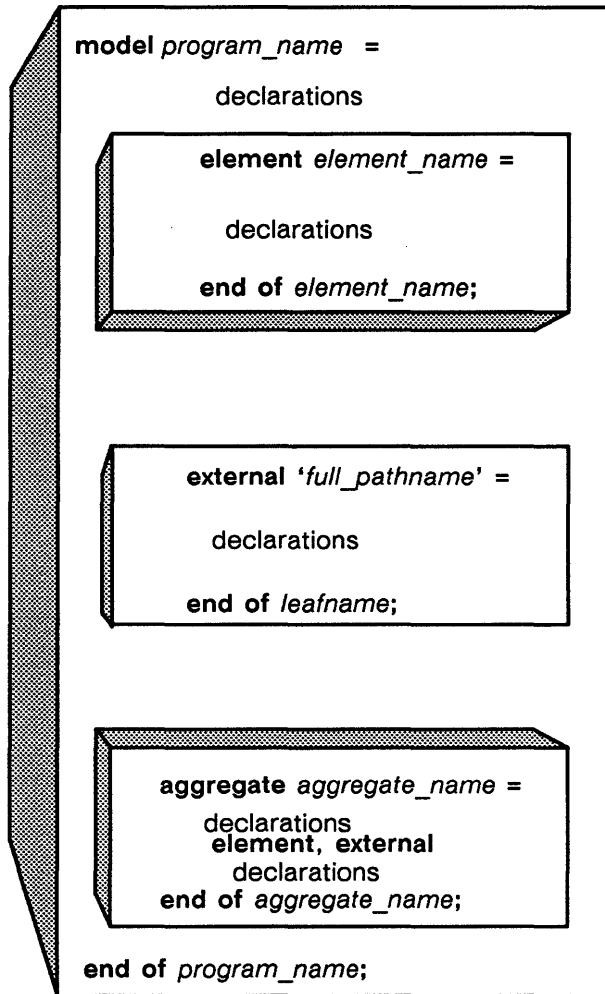


Figure 3-3. Block Structure of a System Model

Block Types

A system model consists of a Model block enclosing any number and combination of Element, External, and Aggregate blocks.

The Model block encompasses the entire program. The Model block contains information that applies to the program as a whole, such as the system directory associated with the model. A Model block starts with the keyword **model** followed by the name of the program (the block ID).

An Element block represents a component whose source file or **primary source dependency** is an element in a DSEE library. An Element block starts with the keyword **element** followed by the name of the element (the block ID).

An External block represents a component whose primary source dependency is a regular file (rather than an element). An External block starts with the keyword **external** followed by the file's pathname enclosed in single or double quotation marks (the block ID).

An Aggregate block has no primary source dependency. Rather, an Aggregate block consists of subordinate Element, External, and/or Aggregate blocks. The components represented by these subordinate blocks may be combined to form a single component, or they may be separate but related components brought together for easy reference within the system model. An Aggregate block starts with the keyword **aggregate** followed by a user-selected block ID.

NOTE: Your sample library contains an element named **order.sml**, which is the system model for the program **order**. We have simplified and annotated this functional system model as an adjunct to the following discussion. To display **order.sml**, use the **read** command on the Version menu.

Declarations

The declarations inside a block typically specify a component's translation rule and direct dependencies.

The **translate** declaration declares a component's **translation rule**, which is the command sequence required to bind, compile, assemble, or otherwise process the component. The **translate** declaration starts with the keyword **translate** and ends with the special symbol **%done**.

A component's **direct dependencies** are its primary source dependency (identified by the block IDs of Element and External blocks) plus any include files, tools, and other components required to build that component. All of a component's direct dependencies must be available before the DSEE facility can build the component. The DSEE facility targets a component for rebuilding if any of its direct or indirect dependencies change.

The declarations that identify a component's direct dependencies are:

depends_source

This declaration generally identifies the include files required by the component's primary source dependency.

depends_tools This optional declaration identifies the tools (such as compilers and binders) used in the component's translation rule.

depends_result

This declaration identifies the subcomponents whose derived objects are required by the component.

When the same declarations apply to more than one block, you can group the declarations together by means of an optional **default** declaration. An unqualified **default** declaration (one that starts with the keyword **default** followed by an equal sign) applies to all sub-blocks of the block containing the **default** declaration. A qualified **default** declaration (one that starts with the keywords **default for** followed by a wildcard specification and an equal sign) applies only to sub-blocks whose block IDs match the wildcard specification.

If an Element or External block has no declarations (for example, you may have declared them in a **default** declaration), then you can represent the block in an abbreviated form. This form consists of the keyword **element** or **external**, the block ID, and a semicolon. The abbreviated forms of the declarations are:

element *element_name*;

external '*full_pathname*';

In addition to the declarations already described, the Model block contains a unique set of declarations for describing the properties of the program as a whole. These declarations include:

alias	This optional declaration declares aliases for the system model.
environment	This optional declaration declares environment variables for the DSEE process and for builds.
host_type	This declaration identifies the type of computer on which an element will be built.
library	This declaration identifies the program's source libraries and assigns them logical names.
shell	This declaration identifies the shell environment where translation rules will be executed.
system	This declaration identifies the system directory associated with the system model.
title	This optional declaration identifies descriptive comments about the program.

Setting the Current System Model

Now that you are familiar with the basic purpose and structure of a system model, set the most recent version of the element **order.sml** as your current system model with the **set model** command.

1. Point to the icon for **order.sml** and click <M1>.
2. Select **set model** from the Model menu.
3. The DSEE facility displays a dialog box seeded with "order.sml." Point to the Confirm button and click <M1>.

The DSEE facility validates a new or modified system model before setting it as your current model. Validation includes checking the model's syntax and semantics, plus verifying that all objects referred to in the model (such as the system, libraries, elements, files, and tools) already exist. If the DSEE facility discovers any errors, it reports them and aborts the command.

At this point, the DSEE desktop interface should resemble Figure 3-4. Note that the context banner displays your current model setting.

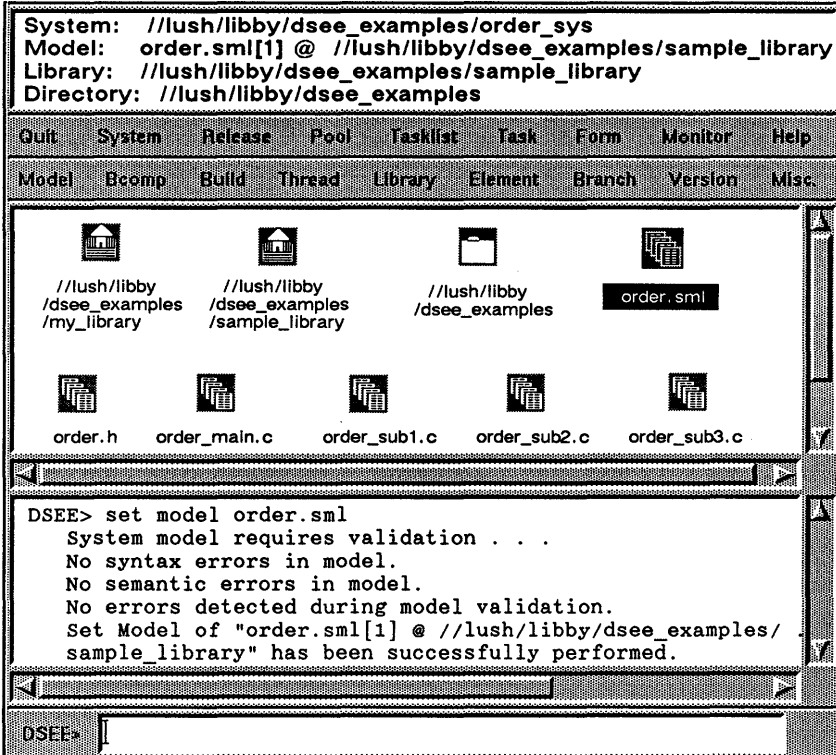


Figure 3-4. set model Command

The Configuration Thread

Having set your current system and current system model, you can now set your current configuration thread.

A configuration thread specifies the element versions and translation options required for a particular build. When the DSEE facility builds a component, it uses the system model to determine the name of the component's source element and the configuration thread to determine the version of that element. The DSEE facility also uses the translation options specified in the configuration thread; these options must appear among the allowable options listed in the model's translation rule for the component.

NOTE: Because files that are not DSEE elements do not have distinct versions, configuration threads do not contain version information about such files.

A **configuration thread** is a list of rules written in the configuration thread language, with each rule occupying a single line in the thread. These rules and their order determine which element versions to use in a build and which translation options to use in a build.

A configuration thread rule can contain one or more of the following parts: a rule qualifier, a version rule, a build option rule, and a build-ID-based rule. How you combine these parts determines the nature of a configuration thread rule; that is, whether a rule applies only to components built from Element blocks, only to source elements, or only to buildable components.

The optional **rule qualifier** determines what components or source elements can be affected by the rule. For example, a configuration thread rule might be limited to source elements. A rule qualifier can limit the rule's applicability even further by specifying one or more particular source elements.

A **version rule** applies to source elements and can specify a version number, a version name, a reserved version, a pathname, or one of several other alternatives.

A **build option rule** applies to buildable components and can specify any translation option allowed by the component's translation rule in the system model. (For example, if the component's translation rule invokes the Pascal compiler, the compiler's `-dbs` option might be a recognized build option.) The DSEE facility uses the options identified by the build option rule when building the component.

A **build-ID-based rule** tells the DSEE facility to use the version specifications and/or build options of an earlier build.

Before building a component, the DSEE facility consults the system model to determine if the build involves a DSEE element and/or a translation rule. If building the component involves an element, then the DSEE facility searches the configuration thread for the first applicable version rule. If the component has a translation rule, then the DSEE facility searches the thread for the first applicable build option rule.

You set and validate your current configuration thread with the **set thread** command and one of its options. If the DSEE facility finds validation errors, it reports the errors and sets the thread. However, you must correct the thread before you can build with it.

The command option `-from` and a pathname tells the DSEE facility to set the thread stored in the specified file.

The command option `-none` tells the DSEE facility to set your current configuration thread to nil.

The command option `-default` tells the DSEE facility to set the default configuration thread as your current thread. The default thread consists of the following configuration thread rules:

```
-reserved  
[]
```

The `-reserved` version rule in the first configuration thread rule directs the DSEE facility to use reserved element versions in your working directory. Because there is no rule qualifier for this rule, the DSEE facility applies the rule to all source elements.

If the DSEE facility does not find a reserved version of a source element in your working directory, it applies the second configuration thread rule. The empty brackets ([]) version rule in this configuration thread rule directs the DSEE facility to use the latest version on the main line of descent of the element.

To set the default thread as your current configuration thread, perform the following steps:

1. Select **set thread** from the Thread menu.
2. When the dialog box appears, point to the button next to the **-default** option and click <M1>.
3. Point to the Confirm button and click <M1>.

Figure 3-5 shows the message that appears in the transcript area after the DSEE facility successfully validates and sets your current configuration thread.

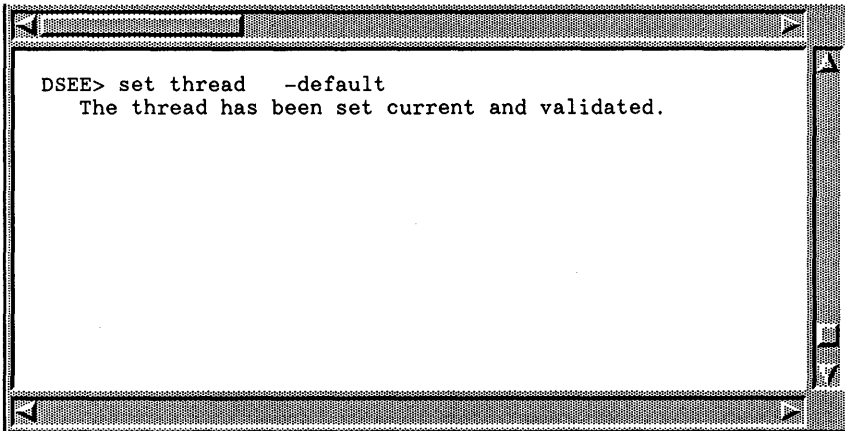


Figure 3-5. set thread Command

Now use the **examine thread** command to display your current configuration thread in the transcript area.

1. Select **examine thread** from the Thread menu.
2. When the dialog box appears, point to the Confirm button and click <M1>.

Your current configuration thread requires one more rule to be compatible with the build procedure later in this chapter. To edit the thread, use the **edit thread** command.

1. Select **edit thread** from the Thread menu.
2. When the dialog box appears, point to the Confirm button and click <M1>.
3. The DSEE facility displays another dialog box. This one contains the text of your current configuration thread (which is the default thread) and asks you to enter the desired configuration thread. Insert the following build option as the first configuration thread rule:

```
-for *.c -use_options -dbs
```

Your thread should read:

```
-for *.c -use_options -dbs  
-reserved  
[]
```

(By default, the **edit thread** command displays the thread using a dialog box. However, you can use the X resource **Dsee*threadEditor** to cause **edit thread** to display the thread using an editor of your choice. The file `/usr/X11/lib/app-defaults/Dsee` lists the X resources for the DSEE software. See the *DSEE Reference* for details.)

The flag `-use_options` in the first configuration thread rule directs the DSEE facility to build with the `-dbs` option. (This option, which tells the compiler to prepare its output for debugging, is allowed because it appears in the system model's default translation rule.) The rule qualifier `-for *.c` indicates that the rule applies to every component whose name ends in `".c"`.

When resolving version specifications, the DSEE facility goes to the first version rule in the thread (**-reserved**). If this version rule does not apply, then the DSEE facility uses the next version rule (**[]**) to resolve a version specification.

4. Point to the Confirm button and click <M1>. The DSEE facility sets and validates the text as your current configuration thread and issues the same message it issued after **set thread** (shown in Figure 3-5).

The Model Thread

There is one setting that you won't need to build this system, but which you should know about. This is the **model thread** setting. A model thread is used when your system model source consists of more than one DSEE element. Like a configuration thread, a model thread identifies the versions of the elements containing model threads that the configuration manager should use when setting the model. (For this reason, you must set your current model thread *before* you set your current model.)

Model threads are also used in conjunction with the system model language's conditional processing feature. In the thread, you identify which predicates you want the configuration manager to use when compiling the system model. These predicates are used to determine the compile-time values of if-then-else statements in the model.

Because your current model contains no if-then-else directives and isn't broken into modules and stored in several elements, you don't need to specify a model thread setting for this exercise.

Initiating a Build

Having set your current system, current system model, and current configuration thread, you can now build your program with the **build** command.

In its simplest form, the **build** command directs the DSEE facility to build the entire program. To build the program **order**, perform the following steps:

1. If the **order.sml** icon is still selected, deselect it (point to an empty space in the icon area and click <M1>).
2. Select **build** from the Bcomp (short for “buildable component”) menu.
3. When the dialog box appears, the text entry box should be empty. (If the text entry box isn’t empty, then delete the text in it.) Point to the Confirm button and click <M1>. The DSEE facility displays a small window during the build that shows the build’s progress.

During the build, the DSEE facility displays a window that contains statistics about the building process, as shown in Figure 3–6. Use the scroll bar at the bottom of the window to display any text that exceeds the width of the window. You can also enlarge the DSEE window to view more text.

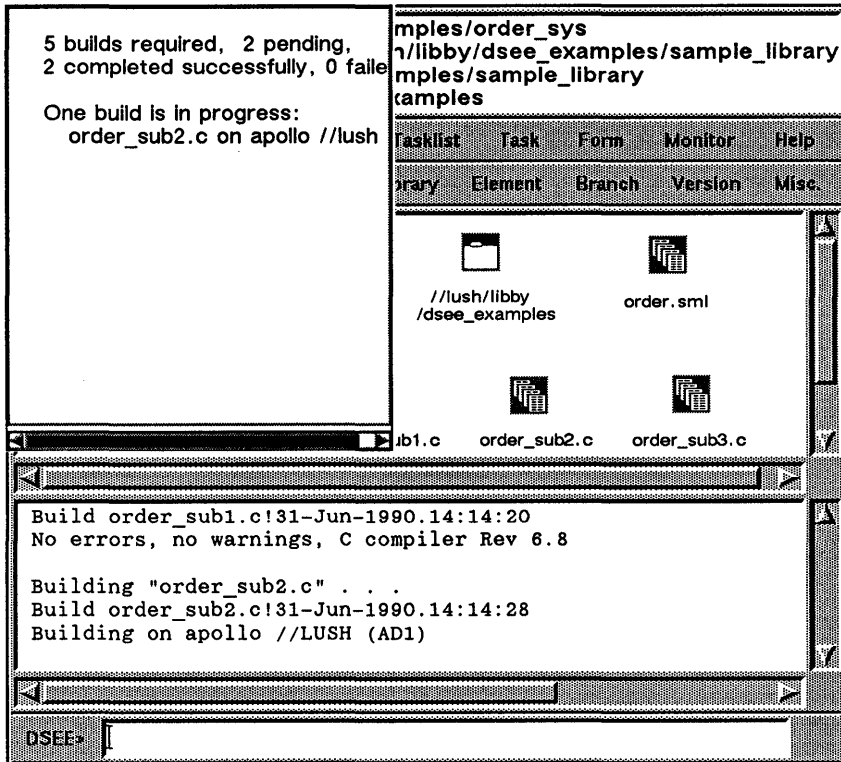


Figure 3-6. build Command

Figure 3-7 shows the messages resulting from a simple, successful build of your program. The first messages displayed in the transcript area pertain to the `.dsee_builder_list` file. The DSEE facility looks for the names of builder nodes in `.dsee_builder_list`. When the DSEE facility can't find a configured `.dsee_builder_list` file, it defaults to performing the build on the node from which the DSEE facility is invoked. (For information on the `.dsee_builder_list` file, see *Using DSEE*.)

Notice that the messages issued by the configuration manager include the name of the node on which each build unit is built. In the next chapter, we show you how to build a system on several nodes concurrently.


```
DSEE> build
The builder config file "/sys/dsee/dsee_config . . .
/.dsee_builder_list" needs to be validated...
?(DSEE) Error in builder config file "/sys/dsee . . .
/dsee_config/.dsee_builder_list": No valid entries found.
(WARNING) The builder config file could NOT be validated.
(WARNING) The builder has been set to -NONE.

No working directory copies of reserved elements were
requested
5 builds are required.

Building "order_main.c" . . .
Build order_main.c!31-Jun-1990.14:14:08
Building on apollo //LUSH (AD1)
No errors, no warnings, C Compiler, Rev 6.8
.
.
Building "order_sub3.c" . . .
Build order_sub3.c!31-Jun-1990.14:14:34
Building on apollo //LUSH (AD1)
No errors, no warnings, C Compiler, Rev 6.8

Building "order" . . .
Build order!31-Jun-1990.14:14:37
Building on apollo //LUSH (AD1)
All Globals are resolved.
```

Figure 3-7. **build** Command

Observe that the DSEE facility displays a build unit specifier and a full build name for every component it builds. In DSEE terminology, a **build unit** is a buildable component and a **build unit specifier** is the ID of a system model block representing a build unit. In Figure 3-7, build unit specifiers appear in boldface type. (The configuration manager counts each build unit that needs to be translated as a “build.” That’s why the messages displayed when you type the **build** command say “5 builds are required.”)

A **full build name** identifies the result of building a component or an entire program (for example, a binary or formatted text file). A full build name in our example consists of a build unit specifier, an exclamation point, and a **build version** (the date and time of the build).

In Figure 3–7, the first four build units are the compiled subcomponents of the program. The last build unit is the executable version of the program created by binding the subcomponents together.

What Happens During a Build

Figure 3–8 illustrates what happens when the DSEE facility builds a component. Before we examine the building process, however, we need to define some more DSEE terms.

Building a component results in one or more **derived objects** (for example, binaries and listing files) and a **bound configuration thread (BCT)**. A BCT specifies which element versions, regular files, tools, and translation rules and options were used in building a derived object. Every derived object has a corresponding BCT.

The DSEE facility stores the derived objects and their BCTs in a reserved area called a **binary pool**. You can create your own binary pool(s) or use the default binary pool in your system directory.

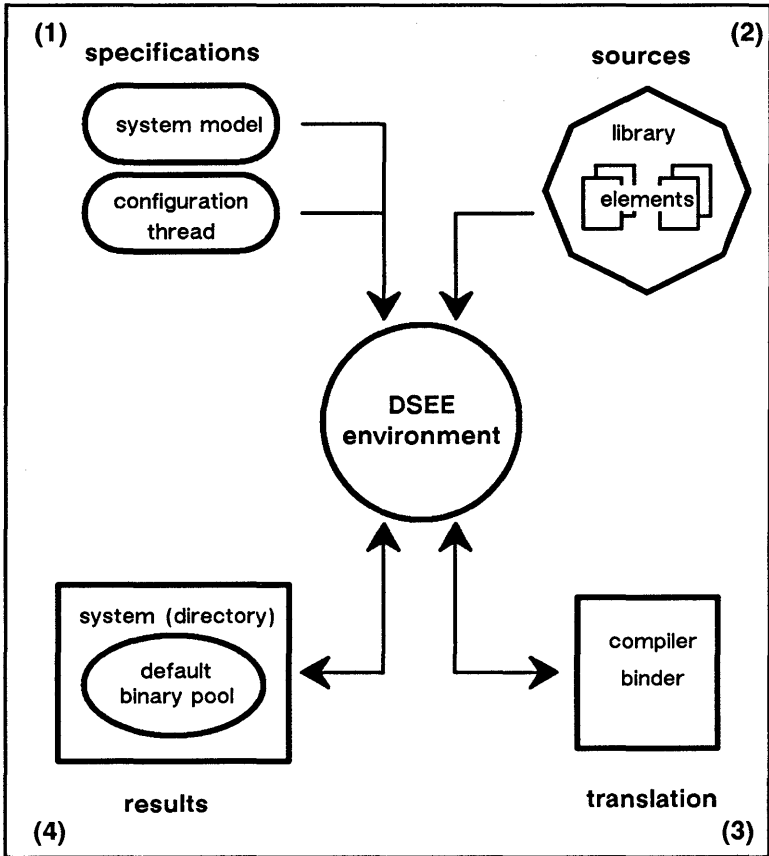


Figure 3-8. Flow of Information During a Build

When the DSEE facility builds a component, it first combines the component's description in the system model, with the version and option information in the configuration thread. The resulting specification, called a **desired BCT**, specifies exactly which element versions, regular files, tools, and translation rules and options are needed to build the component.

The DSEE facility then compares the desired BCT with the BCTs of earlier builds in the binary pool. If a BCT in the binary pool matches the desired BCT, the DSEE facility reuses the derived object associated with that BCT. If there is no matching BCT, the DSEE facility builds a new derived object and BCT and stores them in the binary pool.

A binary pool has limit and age parameters that affect the lifespan of a derived object in the pool. The limit parameter specifies the number of versions of a derived object allowed in the pool. The age parameter specifies the minimum length of time (in hours) that a derived object must stay in the pool before becoming a candidate for automatic deletion. When the number of versions of a derived object exceeds the pool's limit, the DSEE facility deletes the excess on a least-recently-used basis, provided they have reached the minimum age.

Examining a Build

You can examine the build map of a particular build with the **examine build** command. A build map is essentially a readable form of a BCT and includes:

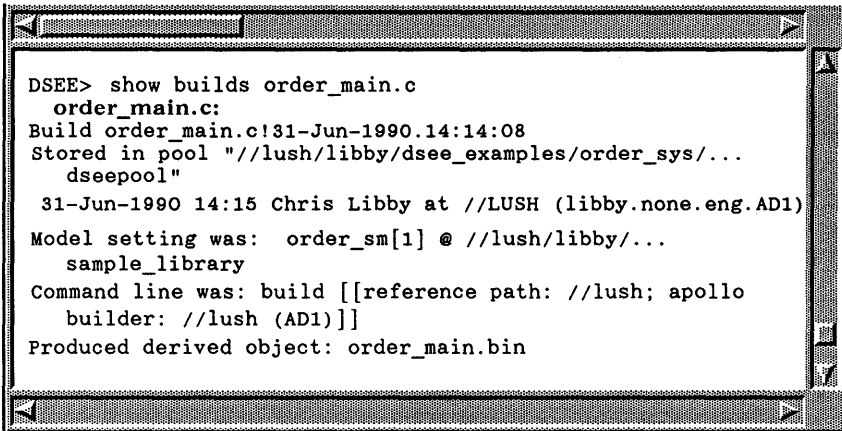
- Date and time of the build (build version)
- System model setting
- **build** command line
- Subject Identifier (SID), node ID, and full name of the person responsible
- Translation rules used in the build
- Element versions and files used in the build

The first four items constitute the build map header.

When issuing the command, you must specify the build's full build name. You can use the **show builds** command to display the full build names of one or more derived objects in your binary pool(s).

1. Point to the icon for **order_main.c** and click <M1>.
2. Select **show builds** from the Bcomp menu.
3. The DSEE facility displays a dialog box seeded with "order_main.c." Point to the Confirm button and click <M1>.

This form of the **show builds** command requests the build map header (including the full build name) of the derived object represented by the build unit **order_main.c**. Figure 3-9 shows the resulting display.



```
DSEE> show builds order_main.c
order_main.c:
Build order_main.c!31-Jun-1990.14:14:08
Stored in pool "//lush/libby/dsee_examples/order_sys/...
dseepool"
  31-Jun-1990 14:15 Chris Libby at //LUSH (libby.none.eng.AD1)
Model setting was: order_sm[1] @ //lush/libby/...
sample_library
Command line was: build [[reference path: //lush; apollo
builder: //lush (AD1)]]
Produced derived object: order_main.bin
```

Figure 3-9. show builds Command

The full build name in your transcript area has a different date and time (build version) than the one in Figure 3-9. The build version shown in Figure 3-9 is

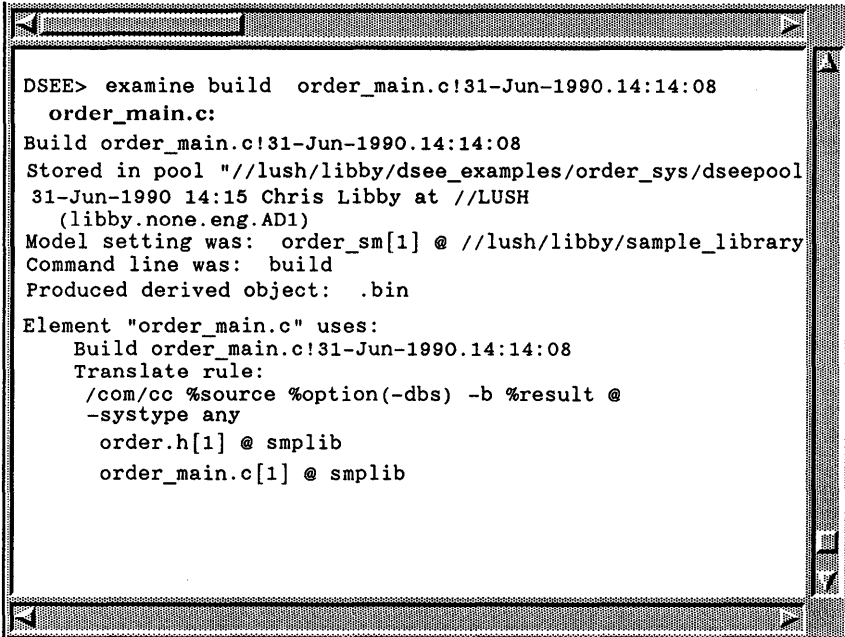
```
order_main.c!31-Jun-1990.14:14:08
```

Use your build version when issuing the **examine build** command below.

1. If the icon for **order_main.c** is still selected, deselect it.
2. Highlight the build version in the transcript area: place the cursor in front of "order," press and hold <M1>, move the cursor to the end of the build version, and release <M1>.
3. Select **examine build** from the Build menu.

4. The DSEE facility displays a dialog box. Paste the build version in the top text entry box: point to the top text entry box and click <M2>.
5. Point to the Confirm button and click <M1>.

Figure 3-10 shows the resulting build map.



```
DSEE> examine build order_main.c!31-Jun-1990.14:14:08
order_main.c:
Build order_main.c!31-Jun-1990.14:14:08
Stored in pool "//lush/libby/dsee_examples/order_sys/dseepool
31-Jun-1990 14:15 Chris Libby at //LUSH
(libby.none.eng.AD1)
Model setting was: order_sm[1] @ //lush/libby/sample_library
Command line was: build
Produced derived object: .bin

Element "order_main.c" uses:
Build order_main.c!31-Jun-1990.14:14:08
Translate rule:
/com/cc %source %option(-dbs) -b %result @
-systype any
order.h[1] @ smplib
order_main.c[1] @ smplib
```

Figure 3-10. examine build Command

Accessing Derived Objects

The derived objects in the binary pools have encoded names that allow the DSEE facility to quickly locate and identify the objects. When you need to access a derived object, you simply specify the object's full build name and the DSEE facility locates the object for you.

Besides having encoded names, derived objects are liable to be deleted from their binary pool. This will happen as successive **build** commands place multiple versions of derived objects in the pool. As the pool gets crowded, the DSEE facility will use the pool's age and limit parameters to delete objects from the pool.

To simplify access to a derived object and to save it from possible deletion, use the **export** command. This command copies derived objects to your working directory or to any specified directory. It can also create links to derived objects in a binary pool for short-term access to those objects.

The **export** command requires the full build name of the derived object. If necessary, you can use the **show builds** command to display this information. You also have the option of using the **Browse** command to display an icon for the full build name. For example, to display an icon for the full build name of the executable version of **order** in your default binary pool, you need to do the following: browse the system model for the build unit (or buildable component) **order**, then browse for the builds associated with **order**.

1. Select **Browse** from the Model menu.
2. From the submenu that appears, select **display direct child buildable components**.
3. The DSEE facility displays a browse box. Select **order** and click on the Confirm button (see Figure 3-11).

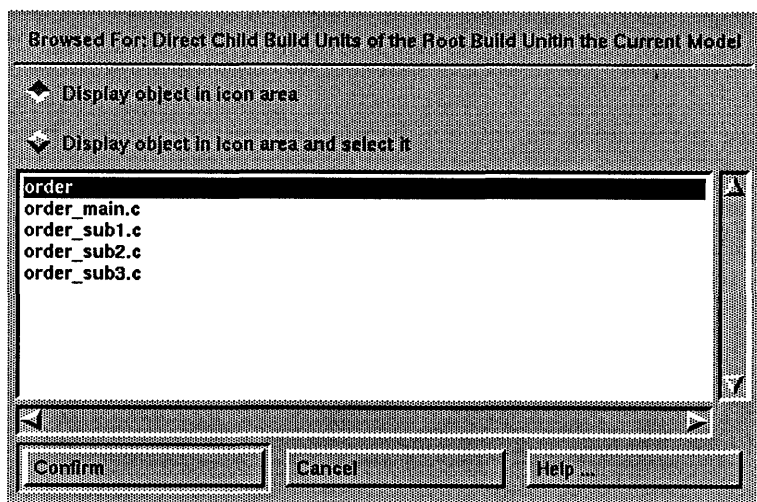


Figure 3-11. Browse Box for Build Units of System Model

4. The DSEE facility displays the build unit icon shown in Figure 3-12. Point to the build unit icon for **order** and click <M1>.



order

Figure 3-12. Build Unit Icon

5. Now browse for the builds associated with the build unit **order**. Follow the menu path shown in Figure 3-13 to select **display all builds** from a submenu of the Bcomp menu.

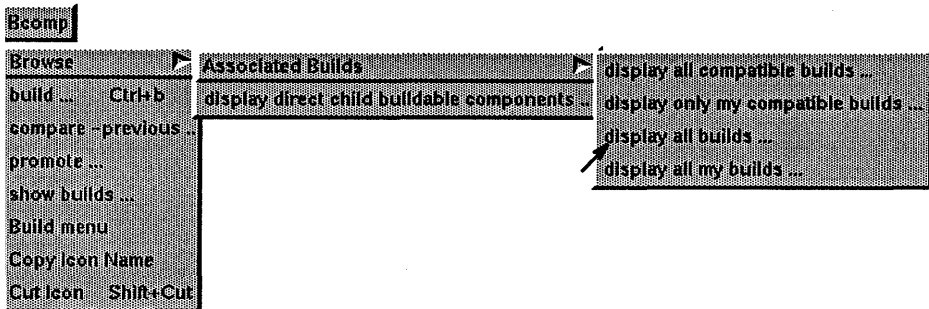


Figure 3-13. Selecting display all builds

6. The DSEE facility displays a browse box containing the full build name for **order**. Point to the full build name and click <M1> to select it.
7. Point to the Confirm button and click <M1>. The DSEE facility displays a build icon for **order** (see Figure 3-14).



Figure 3-14. Build Icon

Now use the **export** command and its **-link** option to create a link in your working directory pointing to **order**.

1. Select the build icon for **order**.
2. Select **export** from the Build menu.
3. The DSEE facility displays a dialog box seeded with the full build name for **order**. Point to the button next to the **-link** option and click <M1>.

4. Point to the Confirm button and click <M1>. The transcript area displays the message:

```
Link "order" created
```

5. To use the link and invoke **order**, enter, in any shell,

```
pathname/dsee_examples/order
```

where *pathname* is the pathname of the directory where you installed **dsee_examples**. You can either respond to the program **order** as directed or abort it by interrupting the process.

Command Summary

The commands discussed in this chapter accept a variety of options. In particular, the options for the **build** command offer a great degree of control over the building process. You should therefore consult complete descriptions of these commands in the *DSEE Reference* before building a system of your own.

Here is a brief list of the commands that you've used in this chapter and their definitions. We group the commands according to the menus from which you can invoke them.

Bcomp

build	Builds the current system model or system component.
show builds	Displays build map headers of one or more derived objects in the current set of pools.

Build**Browse****Associated Builds****display all builds**

Displays builds associated with a build unit. You need to select a build unit icon before issuing this command.

Build

examine build Displays a full build map.

export Exports the derived objects produced from a particular build.

Model

set model Sets the current system model.

Model**Browse****display direct child buildable components**

Displays a list of child buildable components in the current model.

System

set system Sets the current system and sets the current working context.

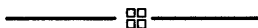
Thread

- edit thread** Edits the text of the current configuration thread.
- examine thread**
 Displays the contents of the current configuration thread.
- set thread** Sets the current configuration thread (or current model thread).

Related Information

System models and configuration threads are explained in great detail in the *Using DSEE*. The manual also contains several examples of system models that you might want to look at.

The next chapter continues investigating the DSEE build process.



Prerequisites to Chapter 4

Before you perform any of the exercises in this chapter, please make sure that you have

- Set your current working directory to the **dsee_examples** subdirectory as directed in Chapter 2.
- Used the DSEE **create library** command to create the **my_library** library as directed in Chapter 2.
- Used the DSEE **set library** command to set your current library to **sample_library** as directed in Chapter 2.
- Used the DSEE **set system** command to set your current system to **order_sys** as directed in Chapter 3.
- Used the DSEE **set model** command to set your current system model to **order.sml** as directed in Chapter 3.
- Used the DSEE **set thread** and **edit thread** commands to set and edit your current configuration thread as directed in Chapter 3.
- Used the DSEE **build** command to build the **order** program as directed in Chapter 3.

Chapter 4

Modifying and Rebuilding a Program

Modifying a program typically involves modifying some but not all of its components. For example, you might create a new source element and edit several others. When you rebuild the program, the DSEE facility then rebuilds every component having a dependency on the new or modified elements. For all other components, the DSEE facility reuses existing derived objects corresponding to those components.

At present, your program consists of four components:

<code>order_main.c</code>	<code>order_sub2.c</code>
<code>order_sub1.c</code>	<code>order_sub3.c</code>

Each of these components has a primary source dependency on the element of the same name and a source dependency on the element `order.h`. Ever since the build in Chapter 3, derived objects corresponding to these components and the program `order` have existed in your default binary pool.

In this chapter, you will modify your program by adding the component `order_sub4.c` and modifying the component `order_main.c`. In the process, you build a single component, declare an equivalence, and rebuild the program using the `build` command and the following new commands and options:

`build -bct_only`

`build -query`

`compare builds`

`set builder`

Modifying a Program

Use the following procedure to create the element `order_sub4.c` and modify the element `order_main.c`.

1. Select the icon for `my_library`.
2. Select `set library` from the Library menu.
3. The DSEE facility displays a dialog box seeded with “`my_library`.” Click on the button next to “library” if it isn’t already selected, then click on the Confirm button.
4. Select `create element` from the Element menu.
5. When the dialog box appears, remove any text in the top text entry box and type

`order_sub4.c`

and click on the Confirm button.

6. The DSEE facility displays another dialog box and requests a comment on the purpose of the element. Type

`Input module`

and click on the Confirm button. A branch icon representing the reserved branch `order_sub4.c` appears in the icon area.

7. Select the icon for **sample_library**.
8. Select **set library** from the Library menu.
9. The DSEE facility displays a dialog box seeded with “**sample_library**.” Point to the Confirm button and click <M1>.
10. Point to the icon for **order_main.c** and click <M1>.
11. Select **reserve** from the Branch menu.
12. The DSEE facility displays a dialog box seeded with “**order_main.c**.” Point to the Confirm button and click <M1>.
13. The DSEE facility displays another dialog box; this one prompts you to enter a comment about the reservation. Type

Add **order_sub4.c**

and click on the Confirm button. A branch icon representing the reserved branch **order_main.c** appears in the icon area.

If you were adding a module to a real system, you would now edit the file **order_main.c** and create the file **order_sub4.c** in your working directory. However, for this example, we've supplied a DSEE command file that does the editing and file creation for you. (We've done this so that you can spend your time learning the DSEE facility, not writing sample code.) Simply run the script **sources** in the **update** directory with the following command. The left angle bracket (<) directs the DSEE facility to use the specified file as input. At the DSEE command line, enter

<update/sources

While the commands in **update/sources** are in progress, the DSEE prompt to the left of the command line fades. When the DSEE prompt no longer appears faded, the commands have completed. (Any time you invoke a DSEE command, the DSEE prompt remains faded while the command is in progress.)

To display edited copies of the reserved elements in your working directory (`order_sub4.c` and `order_main.c`), use the **Edit File** command from the Misc. menu. (The **read** command cannot display reserved elements in your working directory.)

1. Click on the icon for `order_sub4.c`.
2. Select **Edit File** from the Misc. menu.
3. Point to the Confirm button and press <M1>.
4. The DSEE facility displays `order_sub4.c` using the editor the X resource `Dsee*editor` specifies. When you finish viewing `order_sub4.c`, exit from the editor and repeat the procedure to display `order_main.c`.

Do not issue **replace** commands to replace `order_sub4.c` and `order_main.c` at this point. The DSEE facility will leave both elements reserved, with their edited copies in your working directory. This allows you to freely edit the copies when rebuilding and debugging the program. (The DSEE facility uses the copies when rebuilding the program.)

If you were to rebuild the program now using your current system model and current configuration thread, the DSEE facility would (among other things) rebuild the component `order_main.c` and completely ignore `order_sub4.c`. In the first case, your model and thread together would necessitate a rebuild of the component `order_main.c` because you reserved the element `order_main.c` and placed its copy in your working directory. In the latter case, since your model contains no reference to `order_sub4.c`, the DSEE facility would have no reason to build a new component of that name.

Modifying the System Model

Adding a component to your program requires a corresponding change to your system model. You now reserve the element `order.sml`, edit it, and replace it. Again, we've supplied a DSEE command file (`update/system_model`) that will do the editing for you. The following procedure reserves the element, runs the editing script, and replaces a new version of `order.sml` in your current library.

1. Point to the icon for **order.sml** and click <M1>.
2. Select **reserve** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with “order.sml.” Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box; this one prompts you to enter a comment about the reservation.
Type

order_sub4.c

and click on the Confirm button. A branch icon representing the reserved branch **order.sml** appears in the icon area.

5. At the DSEE command line, enter
<update/system_model
6. Select **replace** from the Branch menu.
7. The DSEE facility again displays a dialog box seeded with “order.sml.” If the **-nc** option is selected, deselect it, then point to the Confirm button and click <M1>.
8. The DSEE facility displays another dialog box; this one is seeded with the comment you entered when you reserved the element. Add the following additional comment:

order_sub4.c added

and click on the Confirm button.

Use the **read** command to examine the latest version of your system model.

1. Select **read** from the Version menu.
2. When the dialog box appears, it is seeded with the name “order.sml” (assuming that the **order.sml** icon is still selected). Point to the Confirm button and click <M1>. The DSEE facility displays a read-only copy of the most recent version of **order.sml**.

The model contains two additional lines. The first of these lines,

```
mylib = 'my_library';
```

augments the **library** declaration near the top of the system model. The declaration now identifies two source libraries for the elements specified in the model. Previously, the model identified only **sample_library** and assigned it the library ID **smplib**.

The second line,

```
ELEMENT order_sub4.c @ mylib;
```

is an additional Element block declaration near the bottom of the system model. This abbreviated form of the Element block, like the others above it, represents a component whose primary source dependency is a DSEE element. The element, of course, is **order_sub4.c**. The library specifier **@ mylib** tells the DSEE facility that unlike the other elements in the model, this element resides in the library **my_library**. Notice that placing an explicit library specifier in a block overrides the default library specifier (in the model's first **default** declaration) declared for that block.

Now set the model's most recent version as your current system model.

1. If you haven't closed the **order.sml** file, close it now.
2. Select **set model** from the Model menu.
3. The DSEE facility displays a dialog box seeded with "order.sml" (assuming that the **order.sml** icon is still selected). Point to the Confirm button and click <M1>.

The DSEE facility responds by validating and setting the new system model.

Before rebuilding your program, verify that your current configuration thread contains the correct thread rules for the rebuild.

1. Select **examine thread** from the Thread menu.
2. When the dialog box appears, point to the Confirm button and click <M1>.

The DSEE facility displays the thread in the transcript area. The thread contains the following rules (we've added the line numbers to refer to each rule easily in the text):

```
-for ?*.c -use_options -dbs      (1)
-reserved                        (2)
[]                               (3)
```

Reviewing the thread, you see that the **-reserved** version rule on line 2 requires the DSEE facility to use working directory copies of reserved elements. This version rule, combined with the component information in your current system model, ensures that the desired BCT (bound configuration thread) for **order** will include the working directory copies of the reserved elements **order_main.c** and **order_sub4.c**. For the unreserved elements specified in your model, the version rule on line 3 ensures that the desired BCT for **order** will include the elements' most recent versions in the library.

Performing a Partial Build

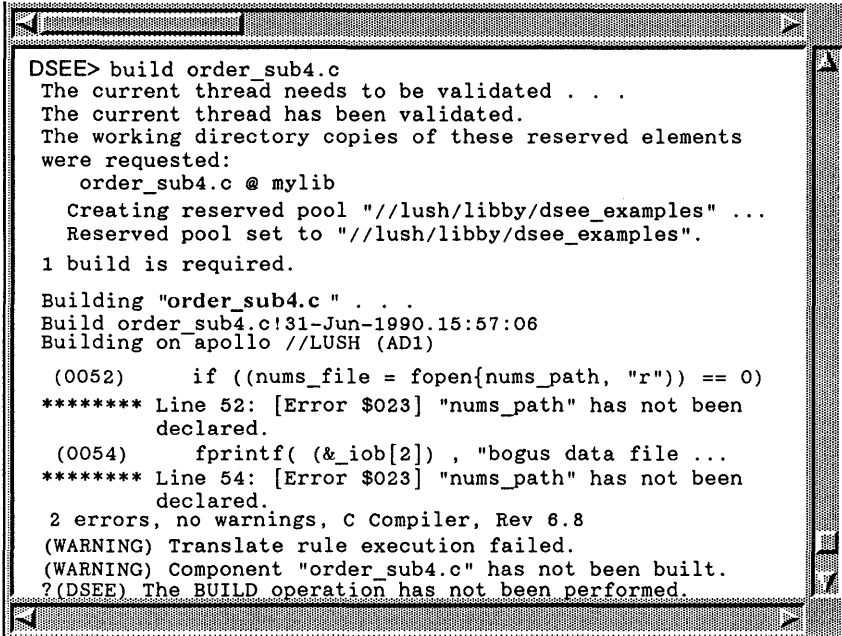
Having modified and set your current system model and reviewed your current configuration thread, you proceed to build only the new component **order_sub4.c**. Your intention is to find and fix compilation errors in the new component before rebuilding the entire program.

Building a Single Component

To build only one component, you issue the **build** command and specify the component's build unit specifier.

1. Point to the icon for **order_sub4.c** and click <M1>.
2. Select **build** from the Bcomp menu.
3. The DSEE facility displays a dialog box seeded with "order_sub4.c." Point to the Confirm button and click <M1>.

Because you have a current configuration thread setting, the DSEE facility validates and resets the thread. Also, the DSEE facility, finding the element `order_sub4.c` reserved, creates a reserved pool in your working directory as indicated in the transcript area shown in Figure 4-1.



```
DSEE> build order_sub4.c
The current thread needs to be validated . . .
The current thread has been validated.
The working directory copies of these reserved elements
were requested:
  order_sub4.c @ mylib
  Creating reserved pool "//lush/libby/dsee_examples" ...
  Reserved pool set to "//lush/libby/dsee_examples".
1 build is required.

Building "order_sub4.c" . . .
Build order_sub4.c!31-Jun-1990.15:57:06
Building on apollo //LUSH (AD1)

(0052)      if ((nums_file = fopen{nums_path, "r"}) == 0)
***** Line 52: [Error $023] "nums_path" has not been
declared.
(0054)      fprintf( (&iob[2]) , "bogus data file ...
***** Line 54: [Error $023] "nums_path" has not been
declared.
2 errors, no warnings, C Compiler, Rev 6.8
(WARNING) Translate rule execution failed.
(WARNING) Component "order_sub4.c" has not been built.
?(DSEE) The BUILD operation has not been performed.
```

Figure 4-1. Unsuccessful Build of Single Component

A reserved pool is for derived objects produced from working directory copies of reserved elements. The DSEE facility automatically creates a reserved pool in your working directory during the build. Derived objects remain in the reserved pool until you replace all of the elements used in building those objects. The DSEE facility then promotes the derived objects by moving them from the reserved pool to the appropriate nonreserved (shared) pool.

Figure 4-1 also shows error messages returned by the C compiler and the DSEE facility. As you can see, the build was unsuccessful because of a missing variable declaration in the component. Because all the components of your program use `order.h` for their common variable declarations, you start your investigation of the problem by looking at the element's latest version.

1. Point to the icon for `order.h` and click <M1>.
2. Select `read` from the Version menu.
3. When the dialog box appears, it is seeded with the name "order.h." Point to the Confirm button and click <M1>. The DSEE facility displays a read-only copy of the most recent version of `order.h`.

Debugging Source Elements

The include file does not contain the new variable declaration required by `order_sub4.c` and `order_main.c`. You therefore reserve the element `order.h` and edit its copy in your working directory. (The DSEE command file edits the copy for you.)

1. If you haven't closed the `order.h` file, close it now.
2. Select `reserve` from the Branch menu.
3. The DSEE facility displays a dialog box seeded with "order.h." Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box; this one prompts you to enter a comment about the reservation. Type

`Add nums_path variable`

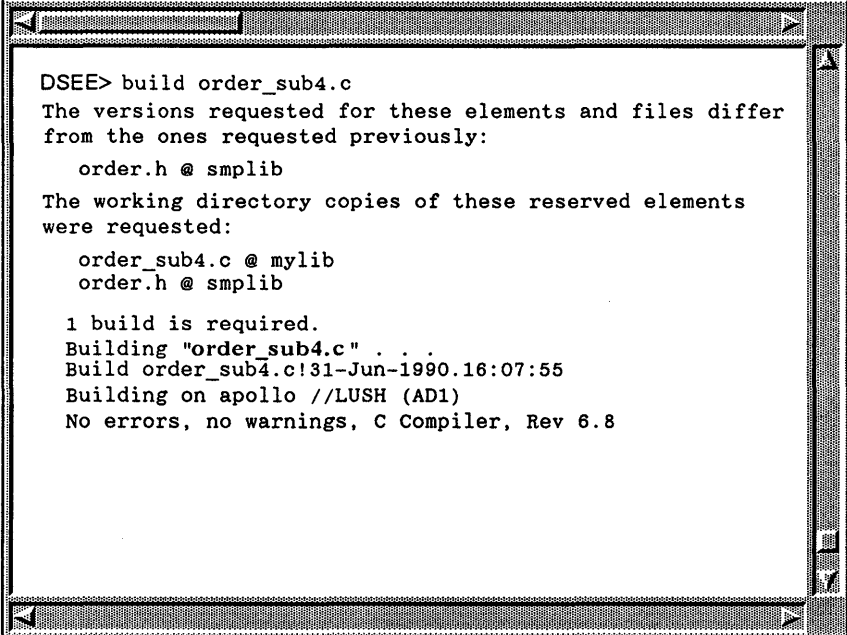
and click on the Confirm button. A branch icon representing the reserved branch `order.h` appears in the icon area.

5. At the DSEE command line, enter
`<update/include_file`

The working directory copy of **order.h** now contains the required variable declaration. (If you want to read the working directory copy of **order.h**, you can use the **Edit File** command. You can see the **Edit File** dialog box by selecting an element, branch, or file icon.) Leaving the element reserved, you try again to build the component **order_sub4.c**.

1. Point to the icon for **order_sub4.c** and click <M1>.
2. Select **build** from the **Bcomp** menu.
3. When the dialog box appears, it is seeded with the name "order_sub4.c." Point to the **Confirm** button and click <M1>.

As indicated in the transcript area shown in Figure 4-2, you succeed. The derived object and BCT built with copies of the reserved elements **order_sub4.c** and **order.h** now reside in the reserved pool in your working directory.

A screenshot of a terminal window with a dark border and a light background. The text inside the terminal is as follows:

```
DSEE> build order_sub4.c
The versions requested for these elements and files differ
from the ones requested previously:
    order.h @ smplib
The working directory copies of these reserved elements
were requested:
    order_sub4.c @ mylib
    order.h @ smplib

1 build is required.
Building "order_sub4.c" . . .
Build order_sub4.c!31-Jun-1990.16:07:55
Building on apollo //LUSH (AD1)
No errors, no warnings, C Compiler, Rev 6.8
```

Figure 4-2. Successful Build of Single Component

Rebuilding the Program

Having successfully built the new component `order_sub4.c`, you prepare to rebuild the program. Your current situation is as follows:

- The current system model declares `order.h` as a source dependency for all components of `order` (as indicated by the `depends_source` declaration in the system model's `default for *.c` declaration)
- The current configuration thread specifies the working directory copy of reserved elements and the most recent version (in the library) of all other elements (as indicated by the last two lines of your configuration thread)
- The elements `order.h`, `order_main.c`, and `order_sub4.c` are reserved (because of `create element` and `reserve` commands you performed earlier in this chapter)
- The derived object corresponding to the component `order_sub4.c` already exists in your reserved pool (because of the last `build` command you issued)

Building on Several Nodes Concurrently

This review of your current situation tells you that all of the components of `order` except `order_sub4.c` will be rebuilt when you issue your next `build` command. In order to speed up your build and take up fewer of your own node's CPU resources, you decide that you want to build the system in parallel on several nodes.

The DSEE facility looks for the names of builder nodes in a file named `.dsee_builder_list`. However, as we describe in this section, you can also specify the names of builder nodes using the `set builder` command. (For information on the `.dsee_builder_list` file, see *Using DSEE*.)

In order to build on several nodes concurrently, issue the **set builder** command before your next **build** command. Using the **set builder** command, you identify:

- The names of all of the possible nodes to be used to build parts of the system (listed in order of most to least preferred). This list can contain as many as 1000 nodes. The configuration manager will use as many as 20 of them concurrently. (See the **limits** help file for current limits.)
- The name of the **reference path**. This is the directory that you want the configuration manager to use to resolve all node-relative pathnames in the system model (for example, system include files).

Issue the **set builder** command as follows:

1. Select **set builder** from the Builder menu (a submenu of the Misc. menu).
2. When the dialog box appears, click on the button next to "Set apollo builders," then click on the Builders button.
3. The DSEE facility displays a second dialog box; this one prompts you to enter a builder list. Enter either names of nodes (for example, //lush) separated by spaces, or the pathname of a file, preceded by an asterisk (*). The named file must contain the names of builder nodes, each one on its own line.
4. When you've finished entering a builder list, point to the Confirm button and click <M1>.
5. Now click on the button next to the **-reference path** option.
6. With the cursor in the dialog box, enter the name of a reference node (your own node, for example).
7. Point to the Confirm button and click <M1>.

Each one of the nodes in the builder list must be running the `server_process_manager` in order to be used as a builder. Issue the BSD shell command `/bin/ps -ax //node_name`, the SysV shell command `/bin/ps -e //node_name`, or the Aegis shell command `/com/pst //node_name` to determine whether a particular node is running the server process manager.

During subsequent builds, the configuration manager will display the name of the node building each component of the system in the transcript. It will also display a pane within the DSEE transcript window that contains statistics of the building process. Figure 4-3 shows an example of a DSEE window during a concurrent build on multiple nodes.

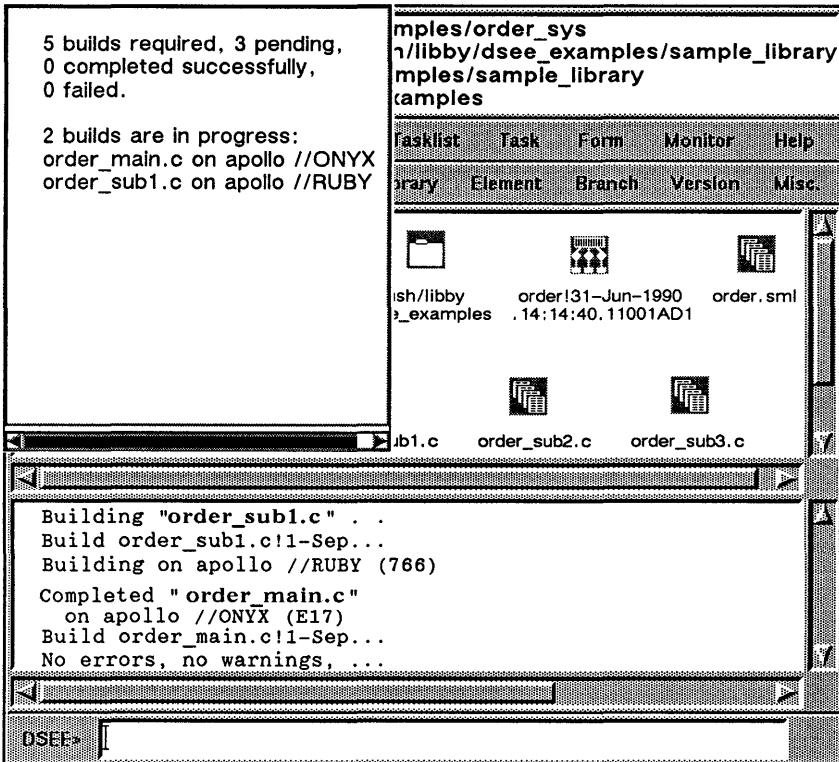


Figure 4-3. DSEE Desktop During Concurrent Build

Figure 4-4 in the next section contains sample output from a **set builder** command and illustrates the display of builder node names during builds.

Your builder settings last until you exit from the DSEE facility.

Rebuilding with Reserved Elements

You now build the modified version of the program **order** with the **build** command.

1. If the **order_sub4.c** icon is still selected, deselect it (point to an empty space in the icon area and click <M1>).
2. Select **build** from the Bcomp menu.
3. When the dialog box appears, delete any text in the text entry box. (You can delete text by first highlighting the text, then pressing <DELETE>. To highlight text, position the cursor at the far right of the text, press and hold <M1>, drag the cursor to the far left of the text, and release <M1>.)
4. Point to the Confirm button and click <M1>.

As indicated in Figure 4-4, the DSEE facility rebuilds all components except **order_sub4.c** (a suitable derived object already existed for this component). In addition, all derived objects produced by the build reside in the reserved pool in your working directory because they each have one or more reserved dependencies.

```
DSEE> set builder //onyx //diamond //ruby -ref //lush
Builders specified in the SET BUILDER command are overriding
the builder list file.
    New builder settings were successfully established for
    host type "apollo".
DSEE> build
The versions requested for these elements and files differ
from the ones requested previously:
    order.h @ smplib
The working directory copies of these reserved elements
were requested:
    order_sub4.c @ mylib
    order.h @ smplib
    order_main.c @ smplib
6 builds are required.
Building "order_main.c" . . .
Build order_main.c!1-Sep-1987.8:27:55
Building on apollo //ONYX (E17)
.
.
.
Completed " order" on apollo //ONYX (E17)
Build order!1-Sep-1987.8.29:32
All Globals are resolved.
Build completed.
    6 builds were required,
    all were completed successfully.
```

Figure 4-4. Rebuild of Entire Program

Testing the Program

To test the program, export a link to **order** and run the program.

1. Select **export** from the Build menu.
2. The DSEE facility displays a dialog box. If you haven't exited DSEE, the dialog box contains the last build unit specifier you entered. Remove the previous specifier, then type

order!

Using only a build unit specifier and an exclamation point (!) for a full build name indicates the last build of that component (for example, **order!**).

3. The button for the `-link` option should still be selected from the last time you issued the `export` command. If not, click on the button now.
4. Point to the button next to the `-r` option and click `<M1>`. The `-r` option directs the DSEE facility to replace an existing link with the new link.
5. Point to the Confirm button and click `<M1>`.

Promoting Derived Objects in a Reserved Pool

Upon testing the program, you determine that none of the reserved elements require modification. You therefore replace all of the reserved elements used in the previous build. As a consequence, you also promote all derived objects produced during that build (that is, they are automatically moved from your reserved binary pool to your default binary pool).

1. Point to the icon for `order_main.c` and click `<M1>`.
2. Select `replace` from the Branch menu.
3. The DSEE facility displays a dialog box seeded with “`order.sml.`” Point to the Confirm button and click `<M1>`.
4. The DSEE facility displays another dialog box; this one is seeded with the comment you entered when you reserved the element. With the cursor in the dialog box, add the following additional comment:

`order_sub4.c added`

and click on the Confirm button.
5. Replace `order.h` in the same way you replaced `order_main.c`, but add the comment

`nums_path added`

instead of “`order_sub4.c added.`”
6. Set your library to `my_library`: Click on the icon for `my_library`, then select `set library` from the Library menu. When the dialog box appears, click on the Confirm button.

7. Now replace `order_sub4.c` in the same way you replaced `order_main.c`, but add the comment

File input

instead of “`order_sub4.c` added.”

8. Now set your library back to `sample_library` in the same way that you set your library to `my_library`.

Notice that the DSEE facility promotes the program’s derived object only after promoting all of its components. If you had deleted any of the program’s subcomponent builds from the reserved pool prior to this time, the DSEE facility would not have promoted the program’s derived object.

Naming Versions Used in a Build

In Chapter 2, you used the `name version` command to assign the version name `sample` to the element versions supplied with this manual. The command can also assign a version name to the element versions used in a particular build, even when the element versions reside in different libraries.

Before you can assign a version name to the element versions used in the last build, you need to browse for its full build name. (The DSEE facility changed the build’s full build name to indicate the date and time that it was promoted from the reserved binary pool to the default binary pool.)

1. In Chapter 3 you browsed for the buildable component `order` and displayed a build unit icon for it. Point to the build unit icon for `order` (see Figure 4–5) and click <M1>.



`order`

Figure 4–5. Build Unit Icon

- Now browse for the builds associated with the build unit **order**. Select **display all builds** from the Bcomp menu (see Figure 4-6).

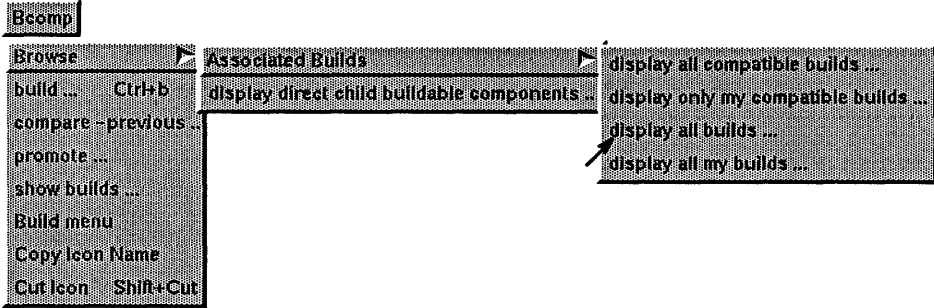


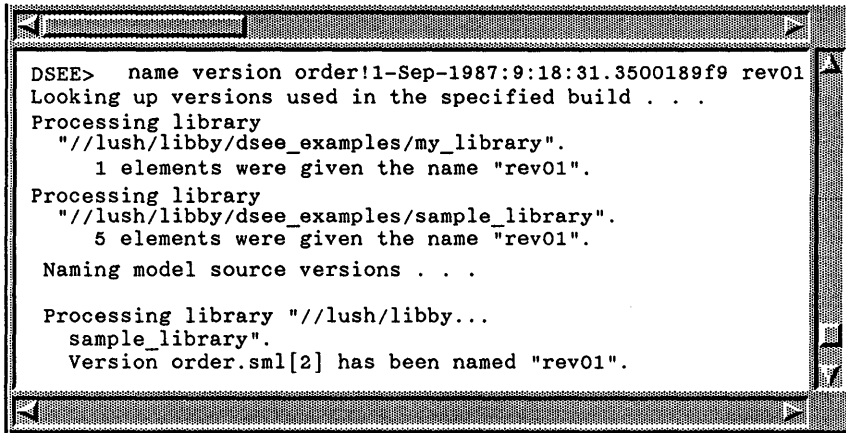
Figure 4-6. Selecting display all builds

- When the browse box appears, select the most recent build (the last build listed).
- Point to the Confirm button and click <M1>. The DSEE facility displays a build icon for the latest build of **order**.

Now assign the version name **rev01** to the element versions used in the last build of **order**.

- Point to the build icon for the most recent build and click <M1>.
- Select **name version** from the Version menu.
- The DSEE facility displays a dialog box seeded with the full build name for **order**. Point to the empty text entry box next to the words "version name:" and click <M1>. Type
rev01
- Point to the Confirm button and click <M1>.

Figure 4-7 shows the messages in the transcript area resulting from the **name version** command.

A screenshot of a terminal window with a dark background and light text. The window has a title bar at the top and a scrollbar on the right side. The text inside the window shows the execution of the 'name version' command in the DSEE environment. The output includes the command itself, a confirmation message, and the processing of two libraries: 'my_library' and 'sample_library'. The 'sample_library' output shows that a version of 'order.sml' has been named 'rev01'.

```
DSEE> name version order!1-Sep-1987:9:18:31.3500189f9 rev01
Looking up versions used in the specified build . . .
Processing library
  "//lush/libby/dsee_examples/my_library".
    1 elements were given the name "rev01".
Processing library
  "//lush/libby/dsee_examples/sample_library".
    5 elements were given the name "rev01".
Naming model source versions . . .

Processing library "//lush/libby...
  sample_library".
  Version order.sml[2] has been named "rev01".
```

Figure 4-7. name version Command and Build

Figure 4-8 illustrates the current state of the element versions in your libraries.

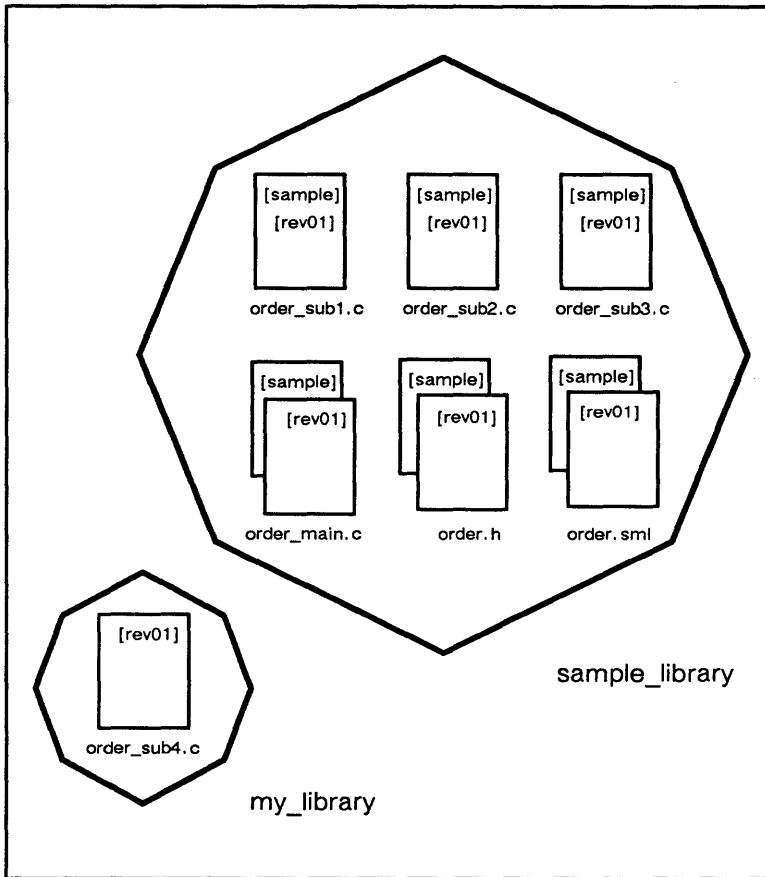


Figure 4-8. Sample Libraries and Their Element Versions

Rebuilding with Existing Components

At this point, you have

- A satisfactory version of the program **order**
- Derived objects in the default binary pool for the program **order** and all of its components
- No reserved elements

As a finishing touch, you decide to add a version identifier to the program. This involves adding a **printf** statement to the element **order_main.c** and a **define** declaration to the element **order.h**. You therefore reserve the elements and edit their copies in your working directory. (As before, we supplied command files that edit these files for you to simplify this tutorial.)

1. Point to the icon for **order_main.c** and click <M1>.
2. Select **reserve** from the Branch menu.
3. The DSEE facility displays a dialog box seeded with "order_main.c." Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box; this one prompts you to enter a comment about the reservation. Type

Adding version ID

and click on the Confirm button.

5. Reserve **order.h** in the same way you reserved **order_main.c**, but add the comment

Adding ID constant

instead of "Adding version ID."

6. At the DSEE command line, enter

`<update/sources_again`

7. At the DSEE command line, enter

`<update/include_file_again`

Use the **Edit File** command from the Misc. menu to examine the reserved files (`order_main.c` and `order.h`) in your working directory.

Having modified the program, you decide to rebuild and test it.

Identifying Components Targeted for Rebuilding

At present, the following conditions exist:

- All of the components defined in your current system model have a direct dependency on the include file `order.h`
- Your current configuration thread specifies the working directory copies of reserved elements
- You have reserved the elements `order_main.c` and `order.h`, and copies of these elements exist in your working directory

To the best of your knowledge, your modifications affect only the component `order_main.c`; the DSEE facility can reuse derived objects for the other components.

To be certain, however, you invoke the **build** command with its **-query** option. This form of the command initiates an interactive build session that begins with a display of the components targeted for rebuilding.

1. Deselect the icon for `order.h`.
2. Select **build** from the Bcomp menu.
3. When the dialog box appears, the text entry box should be empty. Click on the Options button to display all options.

4. Point to the button located to the right of the option, **build querier set to:**, and click <M1>.
5. The DSEE facility displays three more options. Point to the button next to the **-query** option and click <M1>.
6. Point to the Confirm button and click <M1>.

The DSEE facility displays a dialog box that contains the build unit specifiers of components targeted for rebuilding. They are:

```
order_main.c
order_sub1.c
order_sub2.c
order_sub3.c
order_sub4.c
```

7. At this point, you decide that you want to know why the DSEE facility intends to rebuild all these components when your modifications only affected **order_main.c**. You cancel the build command and rebuild the program later.

To cancel the build and close the dialog box, point to the Cancel button and click <M1>. Figure 4-9 shows the resulting messages in the transcript area.

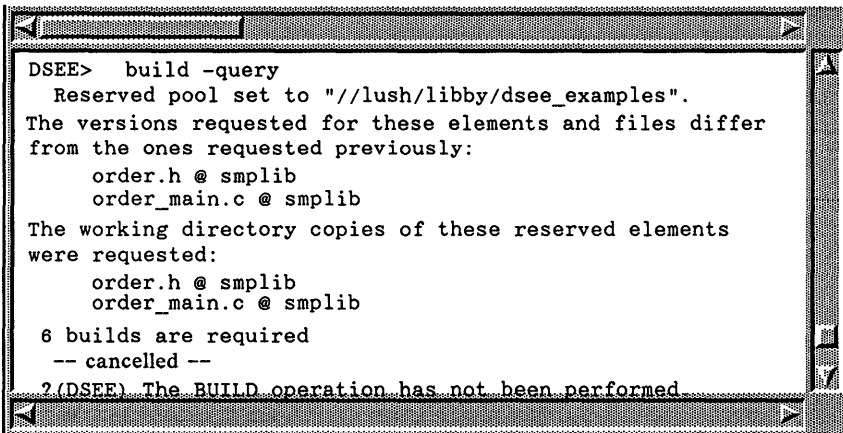


Figure 4-9. Canceled build -query Command

Building BCTs Only

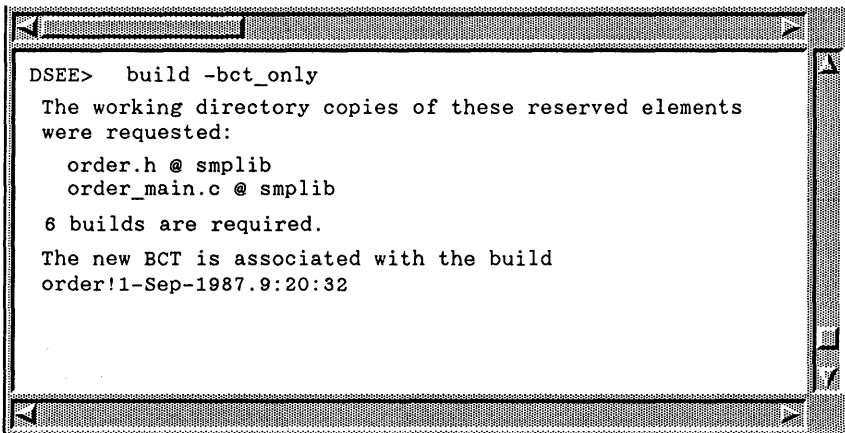
To discover exactly why a component needs rebuilding, you can generate a BCT of the desired build and compare it with the BCT of a previous build. (As you may recall, a BCT contains detailed information about the components of the build, the element versions used in the build, and build options used in the build.)

Normally, building a component produces one or more derived objects and an associated BCT. However, you can request a BCT without a derived object with the **-bct_only** option of the **build** command. This option directs the DSEE facility to produce a BCT without executing any translation rules, and thus without producing derived objects.

Executing a build without translation rules requires much less time than a normal build. The result of such a build is a BCT specifying the exact element versions, files, and derived objects to be used in building the prospective component or program.

As indicated in Figure 4-10, use the **build** command with its **-bct_only** option to create a BCT for the program **order**.

1. Select **build** from the Bcomp menu.
2. When the dialog box appears, deselect the **-query** option: click on the **-query** button, then click on the **no queries** option from the small menu that appears.



```
DSEE> build -bct_only
The working directory copies of these reserved elements
were requested:
  order.h @ smpilib
  order_main.c @ smpilib
6 builds are required.
The new BCT is associated with the build
order!1-Sep-1987.9:20:32
```

Figure 4-10. **build -bct_only** Command

3. Click on the button next to the `-bct_only` option.
4. Point to the Confirm button and click <M1>.

Comparing Builds

The **compare builds** command displays the differences between the build maps of two builds (as mentioned earlier, a build map is a readable version of a BCT).

This command requires two full build names. (Specify the full build name of the older build first to ensure a more readable display.) Browse for the builds associated with **order** to find the full build names of builds in your binary pool.

1. Point to the build unit icon for **order** and click <M1>.
2. Select **display all builds** from the Bcomp menu (see Figure 4-11).

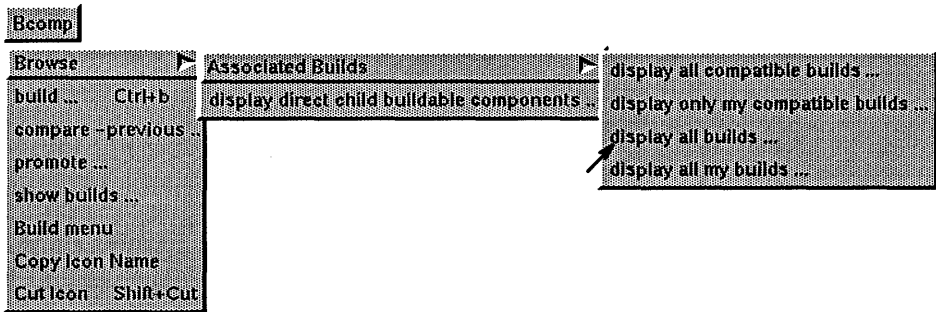


Figure 4-11. Selecting display all builds

3. When the browse box appears, select the most recent build (the last build listed). (You should already have icons for the previous builds displayed in the icon area. If not, then also select the first build in the list, which is the build you performed in Chapter 3.)
4. Point to the Confirm button and click <M1>. The DSEE facility displays a build icon for the latest build of **order**.

Now use the **compare builds** command to display the differences between the BCT you just created and the BCT for the build you performed in Chapter 3.

1. Select the build icon that represents the most recent build.
2. Using the pop-up menu, select **Copy Icon Name**. This command allows you to copy a selected icon name and paste it in dialog boxes or on the command line.
3. Select the build icon that represents the build you performed in Chapter 3 (the build with the oldest date).
4. Select **compare builds** from the Build menu.
5. The DSEE facility displays a dialog box seeded with the name of the older build. Position the cursor in the text entry box that is second from the top and click <M2> to paste the newer build version in the text entry box.
6. Point to the Confirm button and click <M1>.

The build map that you created with the command **build -bct_only** is a representation of what components the DSEE facility thinks it needs to rebuild and why. By looking at the comparison between this new build map and the build map for your earlier build, you see that the DSEE facility thinks it needs to rebuild the components other than **order_main.c** because they require the reserved copy of the include file **order.h** in your working directory. Because the modifications you made to the include file affect only the component **order_main.c**, you decide to reuse the derived objects from the previous build for the remaining components.

Specifying Equivalences

When you want the DSEE facility to reuse an existing derived object instead of building a comparable one, but the facility thinks that it needs to rebuild, you must declare an override or an equivalence. Equivalences and overrides let you explicitly identify older builds to reuse.

An **override** applies only to the current **build** command. When command execution ends, so does the override.

You generally use an override for short-lived builds, such as when you modify a program and rebuild it to test the changes. Then you would probably want to reuse as many derived objects as possible, even though some of the components they represent might need rebuilding.

An **equivalence** can affect other users of the system long after the current **build** command. The DSEE facility stores an equivalence like a BCT, which means that the equivalence can last indefinitely.

You generally use an equivalence when there is no reason to rebuild a particular component now or in the foreseeable future. Because other users of the system might also use the component, this condition should hold true for them as well. If you find yourself declaring equivalences for more than a few components, you should probably isolate your work from that of others on the project by altering your current configuration thread.

There are two types of overrides and equivalences: substitute version specification and substitute build specification. Both types use the configuration thread language.

With **substitute version specification**, you identify substitute versions for some of a component's dependencies. For example, suppose that you tell the DSEE facility not to rebuild the component **order_sub1.c**. You then declare that for this component, element versions **order.h[2]** and **order.h[3]** are equivalent. Rather than rebuild the component with **order.h[3]**, the DSEE facility reuses the component's derived object built with **order.h[2]**.

With **substitute build specification**, you specify the full build name of a substitute build. The DSEE facility then reuses that build instead of rebuilding. (This type of equivalence or override can be destructive because you may unwittingly displace someone else's work. We therefore recommend that you use substitute build specifications only with overrides, which are the default for this type.)

You declare an override or an equivalence during an interactive build session initiated with the **build** command and its **-query** option. As you've already discovered, invoking the command with this option causes the DSEE facility to open a dialog box seeded with the names of components needing to be rebuilt.

Having determined why the DSEE facility targeted all the components for a rebuild, you can now proceed with rebuilding the program **order**. Deciding to declare an equivalence for all components except **order_main.c**, you invoke the **build** command with the **-query** option.

1. Select **build** from the Bcomp menu.
2. When the dialog box appears, the text entry box should still be empty, all options still displayed, and the **-bct_only** option still selected. Deselect the **-bct_only** option.
3. Point to the button located to the right of the option, "build querier set to:", and click <M1>.
4. The DSEE facility displays three more options. Point to the button next to the **-query** option and click <M1>.
5. Point to the Confirm button and click <M1>.
6. Once again, the DSEE facility displays a dialog box containing the following build unit specifiers:

```
order_main.c
order_sub1.c
order_sub2.c
order_sub3.c
order_sub4.c
```

Delete all the names except **order_main.c** and click on the Confirm button. (You can delete text by highlighting the text you want to delete and pressing <DELETE>. To highlight text, press and hold <M1>, drag the cursor across the text, then release <M1>.)

The DSEE facility opens another dialog box for overrides or equivalences involving substitute versions. The dialog box contains a comment and the following identifiers:

```
-EQUIVALENCE
# -override
```

When you insert a version substitution specification in this dialog box, you must accompany it with one of these identifiers. The default identifier is **-equivalence**. To use the **-override** identifier, you must remove the **#** sign that

precedes it and either delete or comment out the **-equivalence** identifier. Because you intend to declare an equivalence, you leave the identifiers untouched.

7. Now use the configuration thread language to write a version substitution specification in the dialog box. For the last line in the dialog box, type

```
-for order.h []
```

This specification directs the DSEE facility to reuse derived objects built with the most recent version of **order.h** (in the library), rather than rebuild the objects using the copy of **order.h** in your working directory. This specification applies only to the components whose names you deleted in the previous dialog box.

The contents of the dialog box are now:

```
-EQUIVALENCE  
# -override  
-for order.h []
```

8. Click on the Confirm button to close the dialog box. The DSEE facility then records the equivalences and rebuilds the program as indicated in Figure 4-12.

Your equivalence declaration makes existing builds of the indicated components equivalent to the desired builds, but only in the reserved pool. Because the build involves equivalences and copies of reserved elements, the DSEE facility stored the equivalences and the derived objects in the reserved pool. When you replace the elements used in the build, the DSEE facility will then promote both the equivalences and the derived objects.

```
DSEE> build -query
The working directory copies of these reserved elements
were requested:
  order_main.c @ smplib
  order.h @ smplib
6 builds are required.
Recording equivalence for "order_sub1.c" . . .
  Build order_sub1.c!1-Sep-1987.9:32:39
Recording equivalence for "order_sub2.c" . . .
  Build order_sub2.c!1-Sep-1987.9:32:40
Recording equivalence for "order_sub3.c" . . .
  Build order_sub3.c!1-Sep-1987.9:32:41
Recording equivalence for "order_sub4.c" . . .
  Build order_sub4.c!1-Sep-1987.9:32:42
2 builds are now required.
Building "order_main.c" . . .
Build order_main.c!1-Sep-1987.9:32:42
Building on apollo //ONYX (E17)
Completed "order_main.c" on apollo //ONYX (E17)
.
.
.
Completed "order" on apollo //ONYX (E17)
Build order!1-Sep-1987.9:33:08
All Globals are resolved.
```

Figure 4-12. `build -query` Command with Equivalences

Now that you have a new version of the program `order`, you export a link to the executable version in the reserved pool and test the program. Use only the build unit specifier `order` and an exclamation point (!) to indicate the last build.

1. Select `export` from the Build menu.
2. The DSEE facility displays a dialog box. With the cursor in the dialog box, type
`order!`
3. If the `-link` option is not selected, select it now.

4. If the `-r` option is not selected, select it now. The `-r` option directs the DSEE facility to replace an existing link with the new link.
5. Point to the Confirm button and click `<M1>`.

You test the new `order` and deem it satisfactory. You therefore replace the reserved elements.

1. Point to the icon for `order_main.c` and click `<M1>`.
2. Select `replace` from the Branch menu.
3. The DSEE facility displays a dialog box seeded with “`order_main.c.`” Point to the Confirm button and click `<M1>`.
4. The DSEE facility displays another dialog box; this one is seeded with the comment you entered when you reserved the element. Add the following additional comment:

Added version ID

and click on the Confirm button.

5. Replace `order.h` in the same way you replaced `order_main.c`, but add the comment

Added ID constant

instead of “Added version ID.”

The DSEE facility replaces the elements and promotes the equivalences and derived objects to the default binary pool.

Rebuilding an Earlier Version of the Program

The time has come to build a version of `order` for wider distribution. You select the earlier `rev01` version of the program because of its relative stability.

When you originally built this version of the program, you assigned the version name `rev01` to the program's constituent element versions. You now want to rebuild the program using only those element versions. You also want to rebuild without the `-dbs` option specified in your current configuration thread.

Editing the Configuration Thread

Your current system model remains unchanged from that previous build. Only your current configuration thread needs editing.

1. Select `edit thread` from the Thread menu.
2. When the dialog box appears, point the Confirm button and click `<M1>`.
3. The DSEE facility displays another dialog box containing the following text:

```
-for ?*.c -use_options -dbs      (1)
-reserved                        (2)
[]                               (3)
```

The thread rule on line 1 requires the `-dbs` translation option when building any component with a build unit specifier ending in `".c"`. Every derived object in your reserved and default pools (except the derived object corresponding to `order`) has been built with this option.

In order to rebuild the entire program `order` without the `-dbs` option, change the thread rule to read:

```
-for ?*.c -exact -use_options
```

Simply removing the `-dbs` option would allow the DSEE facility to reuse derived objects built with that option. The reason for this stems from the translation rule in your current system model. The translation rule looks something like this (on one line):

```
/com/cc %SOURCE -idir . %OPTION(-dbs) -b
                                         %RESULT -systype bsd4.3
```

The `%option` keyword in the translation rule identifies the `-dbs` option as noncritical; that is, the option is not a crucial

attribute of a derived object. This means that the DSEE facility could reuse a derived object built with the `-dbs` option even when you remove it from your configuration thread. Using the `-exact` flag with `-use_options` tells the DSEE facility to build “.c” components with exactly no options.

4. Because you want to build with element versions in the libraries, delete the thread rule on line 2. If you did have any of the elements reserved, this thread rule would specify the wrong version.
5. Also delete the thread rule on line 3. Some of the element versions named `rev01` are previous, not most recent, versions of elements in the libraries.
6. Finally, add a thread rule that reads:

```
[rev01]
```

The version rule `[rev01]` specifies the named version of all elements used in the build (see Figure 4–13).

Normally, you include the `-when_exists` flag with a version rule to tell the DSEE facility to use the specified element version only when that version exists. Without this flag, the DSEE facility reports a thread validation error if a required element lacks the specified version. Because you want to know if a required element lacks a `rev01` version, you therefore omit the `-when_exists` flag.

7. The configuration thread in the edit window should now read:

```
-for ?*.c -exact -use_options  
[rev01]
```

When you’ve entered the text exactly as shown, click on the Confirm button. (Note that the version name `rev01` contains a zero.) The DSEE facility then validates and sets the text as your current configuration thread.

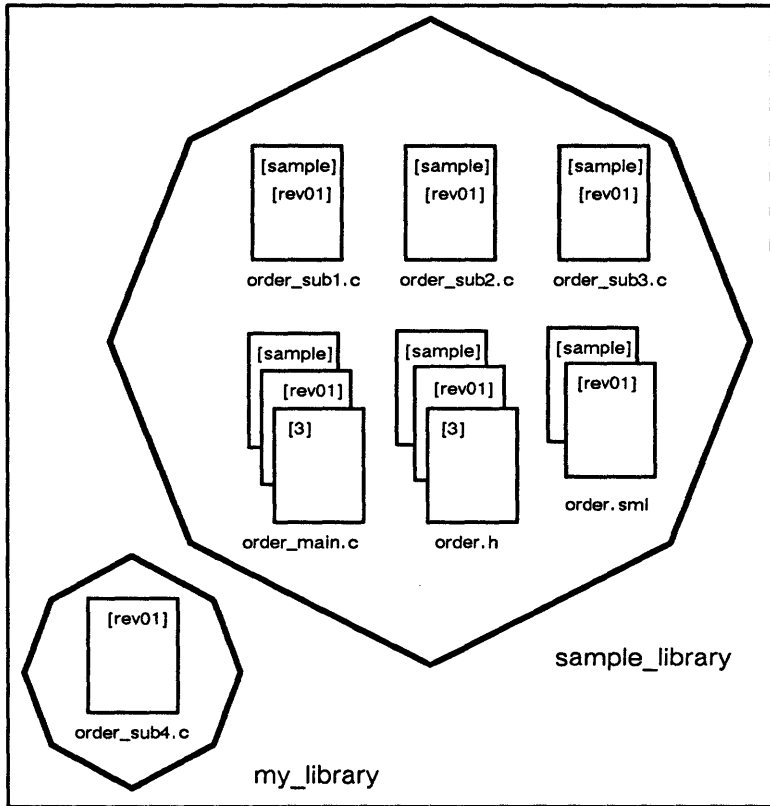


Figure 4-13. Sample Libraries and Their Element Versions

Rebuilding the Program

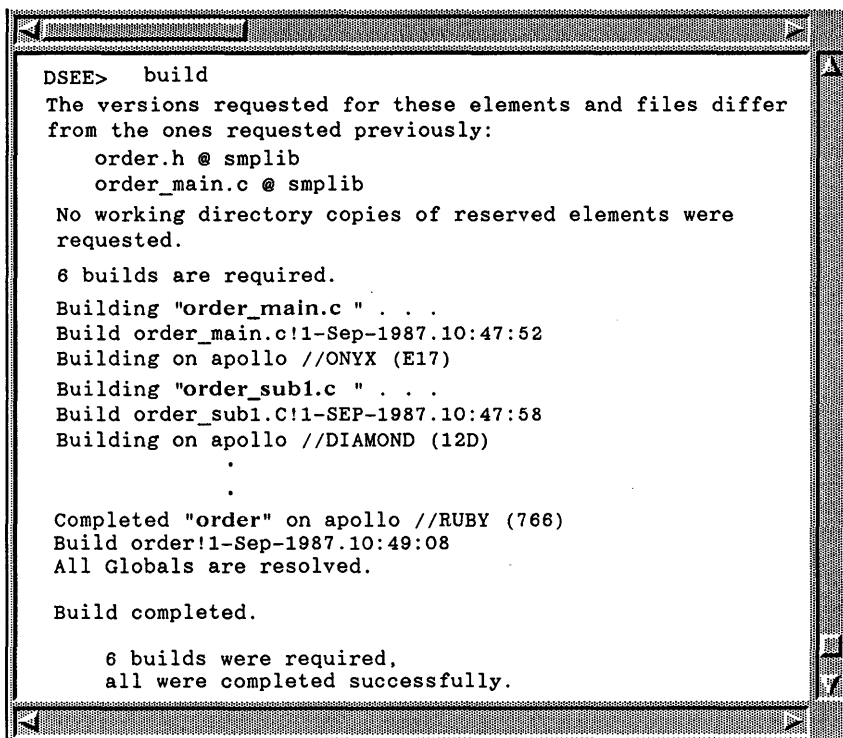
You now rebuild the rev01 version of the program. Because you specified exactly no options, the DSEE facility rebuilds all of the components rather than reuse the derived objects in the default binary pool.

1. If any icons are selected, deselect them.
2. Select **build** from the **Bcomp** menu.
3. When the dialog box appears, the text entry box should be empty. Deselect the **-query** option: click on the **-query**

button, then click on the **no queries** option from the small menu that appears.

4. Point to the Confirm button and click <M1>.

Figure 4-14 shows the resulting messages in the DSEE window.

A screenshot of a window titled 'DSEE' showing the output of a 'build' command. The text is as follows:

```
DSEE> build
The versions requested for these elements and files differ
from the ones requested previously:
  order.h @ smlib
  order_main.c @ smlib
No working directory copies of reserved elements were
requested.

6 builds are required.
Building "order_main.c" . . .
Build order_main.c!1-Sep-1987.10:47:52
Building on apollo //ONYX (E17)
Building "order_sub1.c" . . .
Build order_sub1.C!1-SEP-1987.10:47:58
Building on apollo //DIAMOND (12D)
.
.
Completed "order" on apollo //RUBY (766)
Build order!1-Sep-1987.10:49:08
All Globals are resolved.

Build completed.

6 builds were required,
all were completed successfully.
```

Figure 4-14. Build without -dbs Translation Option

Command Summary

In this chapter you used several variations of the **build** command as well as two other commands. The following list summarizes the commands and variations that you used. We group the commands according to the menus from which you can invoke them.

Build

build Builds the current system model or a system component.

build -bct_only Produces a BCT for a build.

build -query Allows you to identify equivalences or overrides for builds.

Copy Icon Name Copies the name of the selected icon so that you can paste the name in a dialog box or on the command line.

Build

Browse

Associated Builds

display all builds Displays builds associated with a build unit. You need to select a build unit icon before issuing this command.

Build

compare builds Displays the differences between two build maps.

Misc.**Builders**

set builder Specifies one or more build servers, the reference node, and the desired concurrency.

Related Information

This chapter addresses only some of the options available with the **build** command. Discussions of other commands and such concepts as equivalences, overrides, and reserved pools also require more elaboration. You should therefore consult *Using DSEE* for complete descriptions of the concepts and commands mentioned here before building a system of your own.

We list other commands pertaining to configuration management below. We group the commands according to the menus from which you can invoke them.

Build

promote Promotes derived objects stored in a reserved pool to a system pool.

Build**Environment**

create environment

Creates a new shell in which the source reference environment is set to the element version used in a previous build.

delete build Deletes an existing build.

Misc.

Builders

show builders Shows information about build server settings and translation rule execution.

Model

show model Displays information about the current system model.

Pool

configure pool Configures a physical binary pool.

create pool Creates a physical binary pool.

delete pool Deletes a physical binary pool.

purge pool Reclaims space in a binary pool by forcing cleanup.

recover pool Recovers a binary pool.

show pool Displays information about the current system's binary pools.

System

create system Creates a system directory.

delete system Deletes a system directory.

show system Displays the current system setting.

For a comprehensive discussion of all DSEE commands, consult the *DSEE Reference*.



Prerequisites to Chapter 5

Before you perform any of the exercises in this chapter, please make sure that you have

- Set your current working directory to the `dsee_examples` subdirectory as directed in Chapter 2.
- Built, modified, and rebuilt the `order` program as directed in Chapters 3 and 4.

Chapter 5

Managing Releases

In an earlier chapter, we discussed how the DSEE facility automatically purges the contents of binary pools based on the age and limit parameters of those pools. Additional factors such as the number of people using a particular pool and build frequency make it difficult to predict the actual lifespan of a derived object inside a binary pool.

Since your objective in developing, testing, and modifying a program is to release it eventually, you need the ability to move the executable version of a finished program (that is, one or more derived objects in a binary pool) to a stable location. In the DSEE facility, releasing a program involves copying selected derived objects and their BCTs from a binary pool to a release directory with the **create release** command.

Creating a Release

The **create release** command safeguards all or part of a previously built program. When you issue the command, the DSEE facility

- Declares the specified build of the program or its subcomponents to be released and assigns a name to the release

- Creates a release directory
- Copies the release's derived objects and BCTs to the release directory
- Creates a user-readable build map for each derived object in the release

To create a release using your last build session in Chapter 4, perform the following steps:

1. Select **create release** from the **Release** menu.
2. The DSEE environment displays a dialog box. In the top text entry box you need to identify the build session that generated the derived objects and BCTs you want to release. Specify the full build name for your last build session in Chapter 4. Type

`order!`

3. You also need to specify the name of the release and pathname of the release directory. Click on the text entry box next to the words "release directory pathname," then type

`order_rls`

This tells the DSEE facility to create the release directory `order_rls` in your working directory.

4. The button next to the **-export** option is selected by default. The **-export** option identifies the derived objects and BCTs you want to release.

Click on the text entry box for the **-export** option, then type

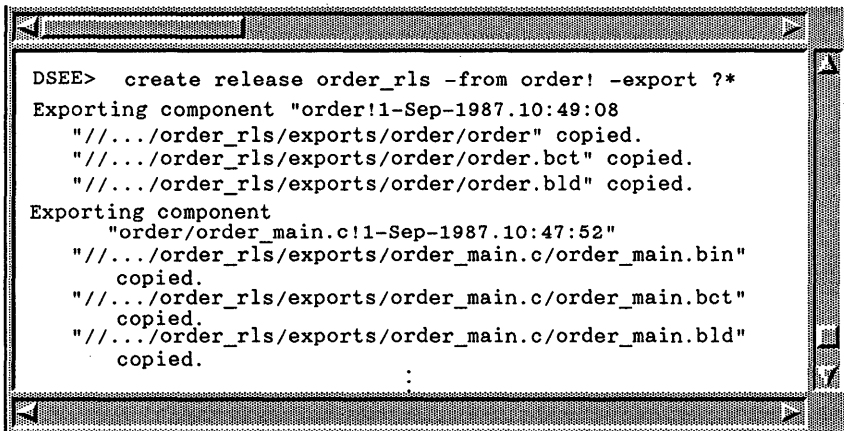
`?*`

This tells the DSEE facility to copy all of the derived objects and BCTs generated during the build (indicated by the wildcards `?*`) into the release directory. Alternatively, you could have specified a list or file of build unit specifiers representing the derived objects and BCTs generated by those build units.

Identifying derived objects and BCTs by their corresponding build units in the system model simplifies the identification process. When you use a build unit specifier, the DSEE facility automatically knows that you mean the derived objects and BCT generated by building that build unit.

5. Point to the Confirm button and click <M1>.
6. The DSEE facility displays another dialog box, seeds it with creation information such as the date and time, and requests a description of the release. Enter a suitable comment (for example, "alpha test release of ORDER") and click on the Confirm button.

Figure 5-1 shows part of the DSEE facility's response to the create release command.



```
DSEE> create release order_rls -from order! -export ?*
Exporting component "order!1-Sep-1987.10:49:08
  "///.../order_rls/exports/order/order" copied.
  "///.../order_rls/exports/order/order.bct" copied.
  "///.../order_rls/exports/order/order.bld" copied.
Exporting component
  "order/order_main.c!1-Sep-1987.10:47:52"
  "///.../order_rls/exports/order_main.c/order_main.bin"
  copied.
  "///.../order_rls/exports/order_main.c/order_main.bct"
  copied.
  "///.../order_rls/exports/order_main.c/order_main.bld"
  copied.
  :
```

Figure 5-1. create release Command

As illustrated in Figure 5-2, the DSEE facility then creates the release directory `order_rls` and a subdirectory named `exports`. Inside `exports`, the DSEE facility also creates one directory for each build unit specified in the `-export` clause of the `create release` command (your command string used `?*` to signify all build units).

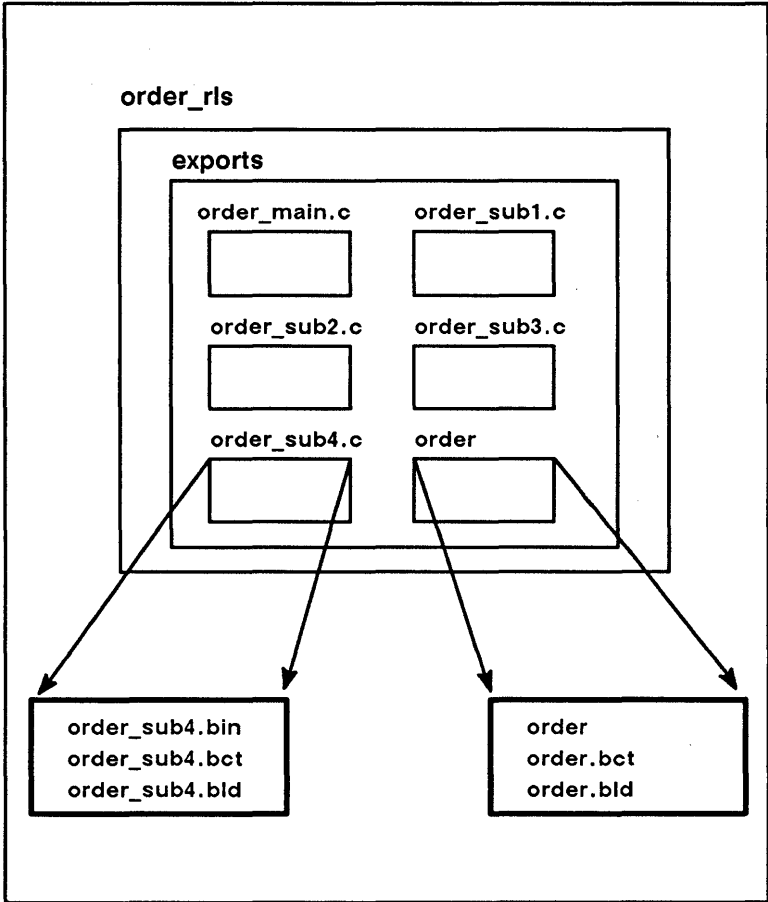


Figure 5-2. Structure of Release Directory **order_rls**

To each build unit's directory, the DSEE facility then copies the corresponding derived objects, BCT, and user-readable build map. Each item copied is assigned the name of the build unit and an extension. If the name of the build unit already ends with a file extension, the DSEE facility strips it off before adding the new extension. The DSEE facility assigns the **.bct** extension to BCTs, the **.bld** extension to user-readable build maps, and whatever extensions apply for the derived objects (**.bin**, **.doc**, **.lpt**, **.lst**, **.txt**, etc.).

Among the subdirectories of `exports` shown in Figure 5-2 is one named `order`. This directory corresponds to the build unit `order` (the Model block in the system model `order_sml`) and therefore contains the executable version of the program, plus the associated BCT and user-readable build map. Thus, the pathname of the executable version of `order` in your release directory is `order_rls/exports/order/order`.

At this point, you may wish to examine the user-readable build maps in your release directory. To do so, use the **Edit File** command or invoke an editor outside the DSEE facility, and specify an appropriate pathname. (For example, specifying the pathname `order_rls/exports/order/order.bld` will display the build map for the entire program.) Each component's build map contains information such as the component's full build name, the system model setting at the time of the build, the build command line, and the applicable translation rule.

Most of the DSEE commands that require full build names (such as the `examine build` and `compare builds` commands) treat a release as any other build. To construct the full build name of a released build, specify the appropriate build unit specifier, an exclamation point (!), and the pathname of the release directory (for example, `order!//lush/libby/dsee_examples/order_rls`). You can also use a released build as a build-ID-based rule in a configuration thread.

Command Summary

In this chapter, we have discussed the most commonly used form of the `create release` command. The command also has options for saving only BCTs and saving the source and/or tool dependencies of a specific build.



create release Creates a release area for built system components.

Related Information

Other commands pertaining to release management are listed below. For details of all DSEE commands, consult the *DSEE Reference*.

Release

- delete release** Deletes a release.
- edit release** Adds to an existing release.
- examine release**
Displays information about a release.
- recover releases**
Updates release area information.
- show releases** Lists releases of the current system.



Prerequisites of Chapter 6

Before you perform any of the exercises in this chapter, please make sure that you have

- Set your current working directory to the `dsee_examples` subdirectory as directed in Chapter 2.

Chapter 6

Managing a Project

Taking software from implementation to release involves many steps, as we've seen in the previous chapters. Some, but not all, of these steps involve changes to DSEE elements. Each of these events is recorded in the history database of the element's library.

The DSEE facility provides a mechanism that helps you organize and record all the steps involved in a process. This mechanism, known as the task manager, lets you define multi-step processes, known as tasks. Tasks act as both "to do" lists and as records of things that have been done. You can assign and organize tasks on lists called tasklists.

Another aspect of the DSEE facility that aids project management is the **monitor manager**. Using the monitor manager, you can monitor one or more elements in a DSEE library. When someone creates a new version of a monitored element, or renames or deletes a monitored element, your monitor is activated. When activated, a monitor can create a new task and add it to tasklists. A monitor can also execute shell commands; for example, it can send mail notifying team members of a change to an element.

The commands discussed in this chapter are:

create monitor

create task

delete monitor

edit task

examine task

examine tlist

set tlist

Before you begin, we suggest that you cut all icons from your icon area other than the library icons and the directory icon. You do not need the other icons to perform the exercises in this chapter.

Using Tasks and Tasklists

The **create task** command enables you to create a task. A task is an object that describes the steps required to accomplish some specific function, such as fixing a bug in a program or adding a new module. When you issue a **create task** command, the DSEE facility stores your new task in your current library.

Your current task setting is part of your working context, like your current system and current library settings. The **create task** command will set your current task, if you wish.

A task has three parts:

- A title (for example, “Fix bug reported in User Report 99”).
- A list of active items (that is, a “to do” list, or the steps required to accomplish the task that have yet to be executed).
- A transcript, or list of completed items. This list contains items moved from the list of active items once they’ve been executed.

The task transcript also contains element change records, which the DSEE facility adds automatically. These element change records describe certain DSEE procedures performed while the task is set as the current task.

These three parts of a task identify the key advantages of tasks. The title allows you to identify the goal of your work (for example, “Get new product ready for shipment”). The list of active items lets you plan your work. The transcript lets you record events related to your work.

Tasks are organized on **tasklists**. You can use a tasklist to identify all the tasks related to one person, one project, one group of people, one deadline, etc. A task can be listed on any number of tasklists.

Like your current task, your current tasklist is part of your working context. More than one person can share the same current task or current tasklist setting at one time.

You can create a tasklist with the **create tlist** command. In addition, the DSEE facility automatically creates three types of tasklists: one personal tasklist for each user, one active tasklist for each library, and one master tasklist for each library. The library tasklists are generally used for record-keeping.

Your personal tasklist serves as a list of all the tasks you personally have to perform, so you will probably use it more often than any other type of tasklist. You refer to your personal tasklist with the syntax “*^userid*” (for example, *^libby*).

Creating a Task

You generate the parts of tasks with the DSEE task editor, a menu-based facility that is tailored for use with tasks. You can invoke the task editor as part of the **create task** procedure.

Here is a hypothetical situation that will guide you through creating and editing a task.

Now that you’ve learned how to use the DSEE facility to manage sources and build systems, suppose that your manager asks you to write up a document telling other new people how to use the DSEE facility. Your manager wants this document to be the first in a group of documents on project procedures and protocol.

Create a task that describes the activities involved in writing up this new documentation.

1. Select **create task** from the Task menu.
2. Click on the button next to the **-title** option.
3. In the text entry box next to “task title text:” type
Document DSEE learning procedure.
4. Point to the Confirm button and click <M1>.
5. In the transcript area, the DSEE facility tells you it has stored the task and added it to your personal tasklist. The DSEE facility also displays a query box that asks whether you want to set your new task as your current task.

Point to the Yes button and click <M1>. The name of the new task is then added to the context banner area.
6. Next, the DSEE facility displays a query box that asks whether you want to edit the task. Point to the Yes button and click <M1>. The task editor takes control of the DSEE window and displays the contents of the task.

Figure 6-1 shows the process of creating a task.

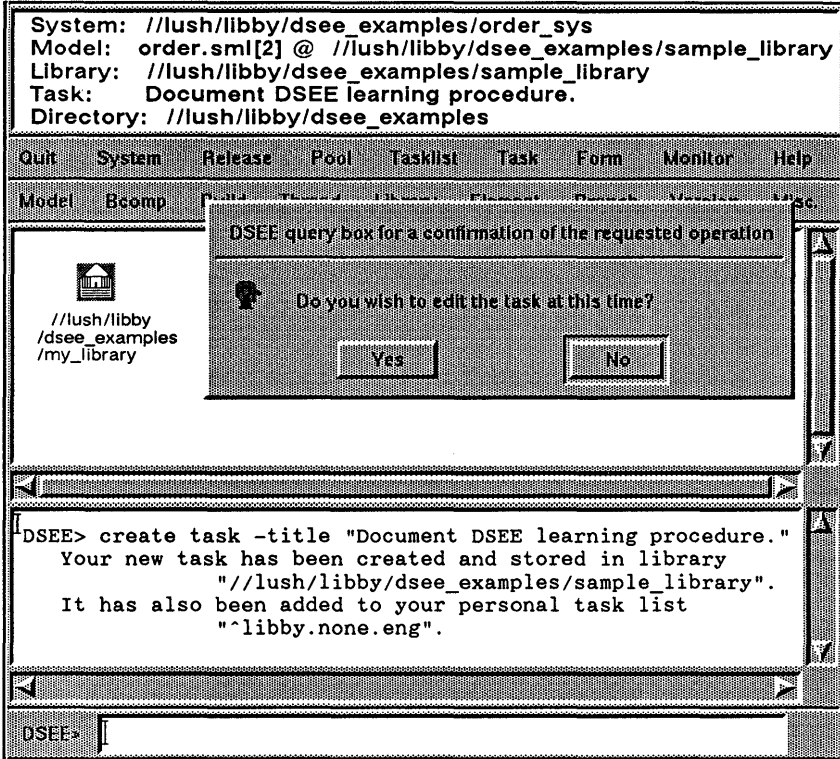


Figure 6-1. Creating a Task

Editing a Task

As we mentioned above, the task editor is menu-based and tailored for use with tasks. It is composed of three main parts, as shown in Figure 6-2.

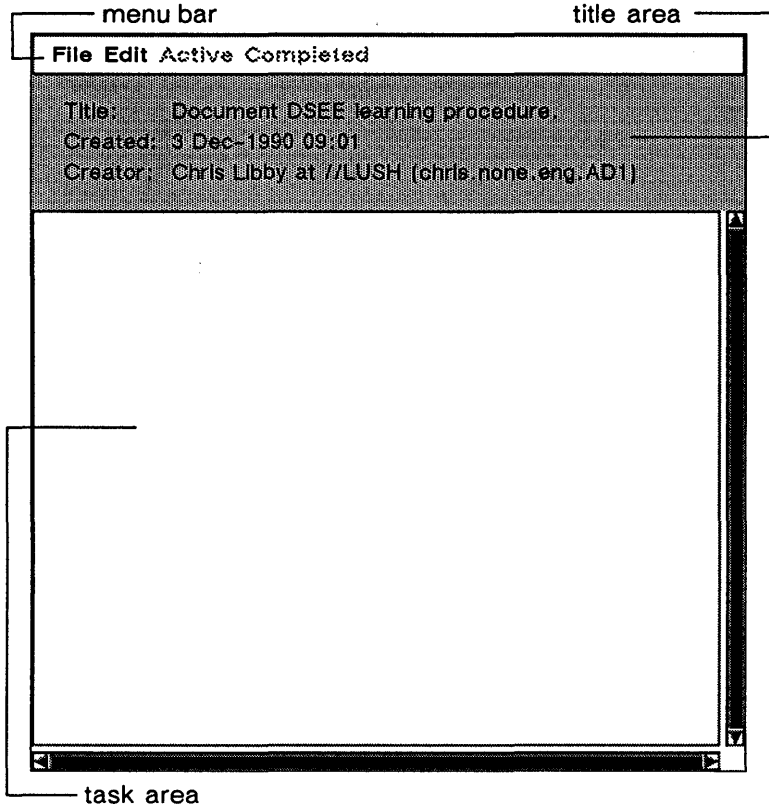


Figure 6-2. Task Editor

The title area displays the task's title, the date and time the task was created, and the task's creator. So far, these are the entire contents of the task.

The task area displays a list of active items and completed items. You will add items to the task area below.

The menu bar offers four pull-down menus, two of which are disabled (as indicated by the faded lettering of the Active and Completed menu titles). You must select an active item in the task area to use the Active menu or a completed item to use the Completed menu.

The steps required to document the DSEE learning procedure are the steps that should appear in the task's active items list. The first

entry on your list of active items should be the first step that you need to take to achieve the goal of the task. In this example, the first step that you'll need to take is to create the new library that will hold the project's process documentation.

To create an active item, perform the following steps:

1. Select **enter new item** from the Edit menu.
2. The task editor displays a dialog box that asks you to type in the text of the new active item. Type

Create library to hold project procedure documents.
3. Point to the Confirm button and click <M1>. The task editor places your active item in the task.

You can also create an active item using the Active menu.

1. Point to the text of the active item you just created and click <M1>. The task editor highlights the active item and enables the Active menu.
2. Select **enter new item** from the Active menu.
3. The task editor displays a dialog box that asks you to type in the text of the new active item. Type

Create an element in the new library for the DSEE learning document.
4. Point to the Confirm button and click <M1>.

The task editor places your new item at the top of the list of active items. To move it to the bottom, perform the following steps:

1. Point to the text of the first active item on the list and click <M1>. The task editor highlights the active item and again enables the Active menu.
2. Select **change priority of item** from the Active menu. (Changing the priority of any active item other than the first one moves the item to the top of the list. Changing the priority of the active item at the top of the list moves it to the bottom.)

Now create a list of active items:

3. Add new items to the list and change their priority until your task looks like the one in Figure 6-3. When you add a new item, it appears directly above the active item you selected before adding the item.

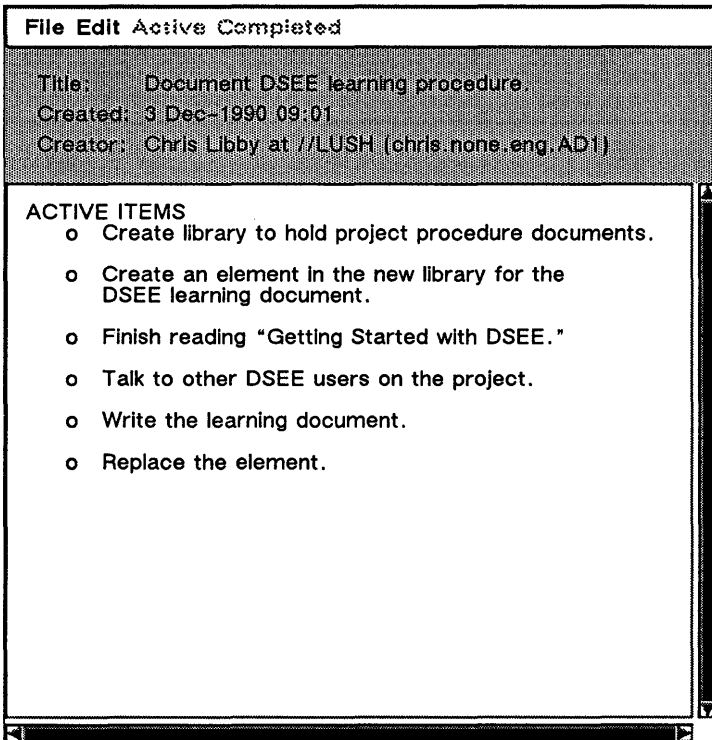


Figure 6-3. Editing a Task

4. Experiment a little by selecting some of the other choices on the active item menu. When you're done experimenting, leave your task looking like the one in Figure 6-3.
5. Select **exit** from the File menu. The task editor writes the contents of the task out to its storage area. (The **abort** selection on the File menu ends the edit without saving your changes to the task.)

Examining Tasklists

When you issued the **create task** command above, the DSEE facility issued a message saying that it had added the task to your personal tasklist. You should look at your personal tasklist now to see the result. In order to do so, you must set your personal tasklist as your current tasklist.

1. Select **set tlist** from the Tasklist menu.
2. The DSEE facility displays a dialog box. In the text entry box, type

^userid

where *userid* is your userid. (The caret (^) identifies this as a personal tasklist, for example, ^libby.)
3. Point to the Confirm button and click <M1>.

Now examine your tasklist with the **examine tlist** command.

1. Select **examine tlist** from the Tasklist menu.
2. The DSEE facility displays a dialog box. Point to the Confirm button and click <M1>. (By not specifying a task list specifier in the text entry box you are indicating that you want to display all the tasks in the tasklist.)

Figure 6–4 illustrates the results. (Note that your personal tasklist’s name is added to the context banner area.)

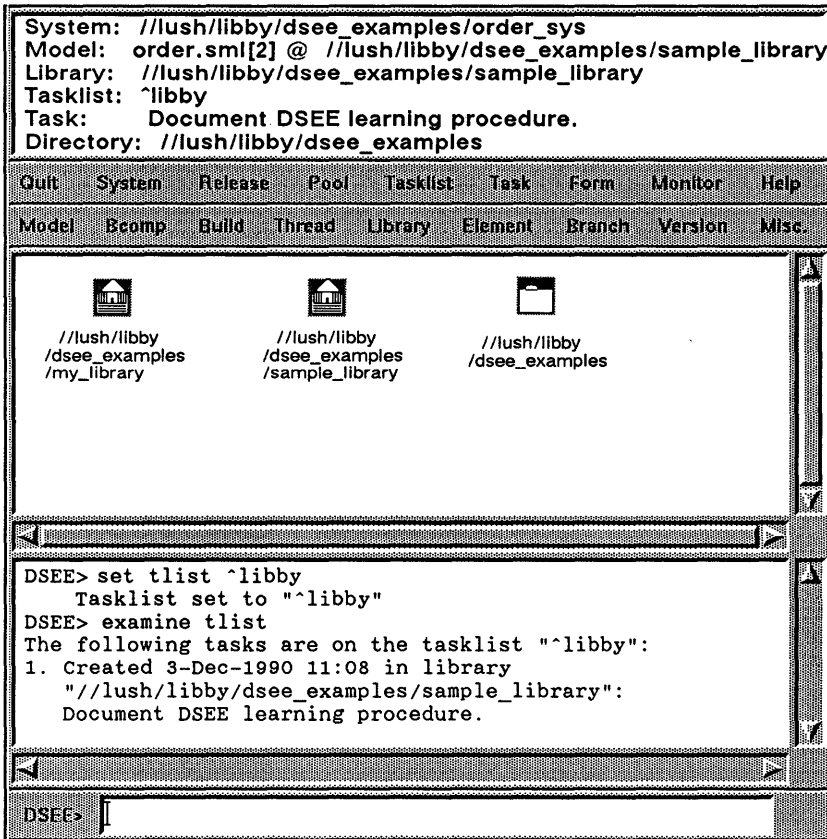


Figure 6-4. Examining a Tasklist

Every task on a tasklist is assigned a number. (In our example, the task "Document DSEE learning procedure." is task number 1 on your personal tasklist.) Whenever you want to refer to any task other than the current task (for example, in commands like **delete task**), you use the task's number on the current tasklist.

Recording Events in Tasks

As we mentioned earlier in this chapter, tasks serve as records of events completed as well as lists of things to do. The component of a task that provides its record-keeping abilities is the **task transcript** (or list of completed items).

When you looked at the menu of operations to perform on an active item, you may have noticed the choice **check off item**. If you selected this choice, the task editor would move the active item into the task transcript, marking it as completed.

Perform the first two steps on the active item list of your current task. The first is to create a library to hold all of the documentation on project procedures and protocol.

1. Select **create library** from the Library menu.
2. The DSEE facility displays a dialog box. In the text entry box, type

project_doc

and click on the confirm button.
3. The DSEE facility displays another dialog box. This one asks you to enter the function of the new library. Type

Documentation library
4. Point to the Confirm button and click <M1>.

Once you've created the library, you can edit the task and move the first item on the list of active items to the transcript. Use the **edit task** command to do this.

1. Select **edit task** from the Task menu.
2. The DSEE facility displays a dialog box. In the top text entry box, type

1
3. Point to the Confirm button and click <M1>.

(Notice that you can edit the task even though it's not stored in the current library. In general, where a task is stored has no effect on how you use it.)

4. When the task editor displays the task, point to the active item “Create library to hold project procedure documents” and click <M1>.
5. Select **check off item** from the Active menu.

You can also record items in the transcript that have not appeared on the list of active items. If, for example, you want to keep a note to yourself about a conversation you had with your manager about the new project procedure documentation library, you can record it in the task transcript. To record an item in the task transcript, you can use the Edit menu, or you can use the transcript menu as follows:

1. Point to the completed item in the task transcript and click <M1>. This enables the Completed menu.
2. Select **enter transcript item** from the Completed menu.
3. The DSEE facility displays a dialog box that asks you to enter the text of the item. Type in a comment, such as

Spoke to manager on 3/4. Manager’s thoughts on project procedure organization are in /update/notes.
4. Point to the Confirm button and click <M1>. The task editor places the new item in the transcript.

Figure 6–5 illustrates the task at this point.

5. Select **exit** from the File menu.

Tasks are integrated with the history facility. When you create a new version of an element, the DSEE facility records the event in the transcript of your current task. It also records the name of the current task in the history of the element version. This cross-reference helps you associate work in progress (in this case, version creation) with the tasks that require it.

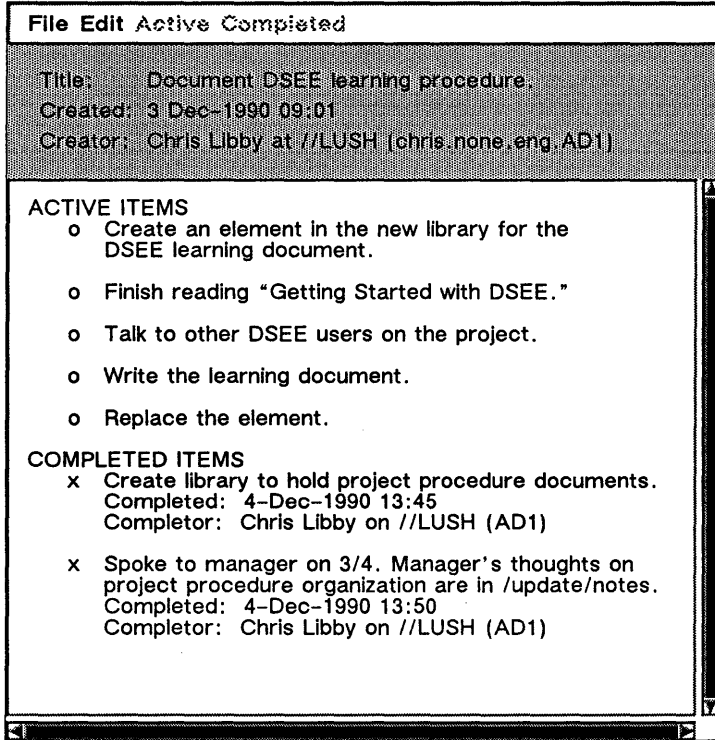


Figure 6-5. Checking off an Active Item

Your manager now wants you to put the notes on project procedure organization into the new library that you created. For this you would use the **-from** option to the **create element** command. The **-from** option tells the DSEE facility to create your new element using the contents of a specified file as the element's first version.

1. Select **create element** from the Element menu.
2. The DSEE facility displays a dialog box. In the top text entry box, type

`mgr_note`

3. Click on the **-from** option. In the second text entry box, type

update/notes
4. Point to the Confirm button and click <M1>.
5. The DSEE facility displays another dialog box. This one asks you to describe the purpose of the element. Type

Element to contain manager's notes on project procedure organization.
6. Point to the Confirm button and click <M1>.
7. When the DSEE facility displays the next dialog box in which you are to describe the initial version of the element, you'll notice that the dialog box already contains text, as shown in Figure 6-6.

You can edit this text, or you can supplement it with more information on the initial version. Because we want to keep the information about the current task in the element's history, do not change the text the DSEE facility has supplied. Click <M1> on the line below the supplied text and type

Original document from manager.

8. Point to the Confirm button and click <M1>.

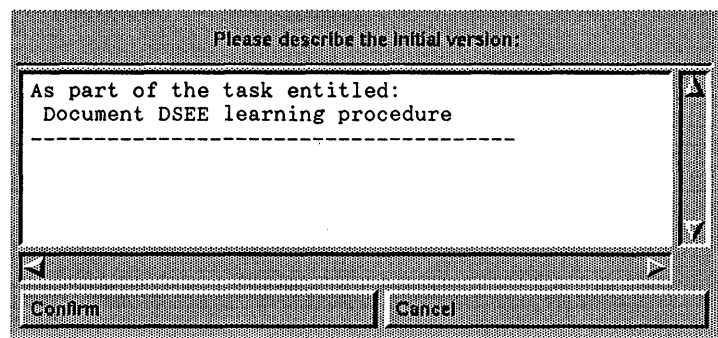
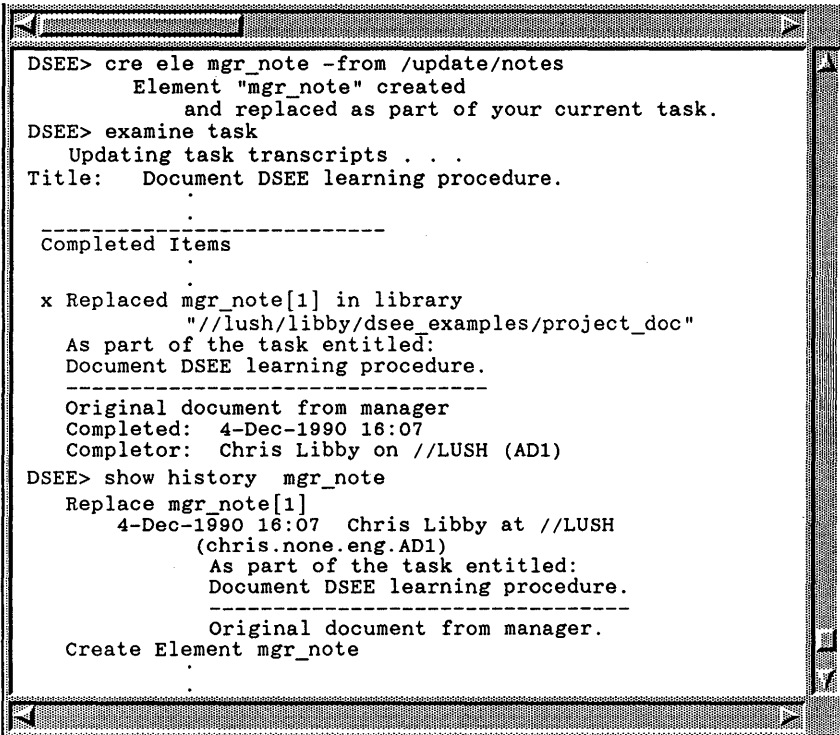


Figure 6-6. A Task Is Noted in Element History

Once you've created the element and the initial version, issue the `examine task` (which displays a task) and `show history` commands to see what has happened. Figure 6-7 contains an example. Notice that the task transcript contains a new entry for the first version of the new element. This entry includes the comment you provided as well as other information about the event.



```
DSEE> cre ele mgr_note -from /update/notes
      Element "mgr_note" created
      and replaced as part of your current task.
DSEE> examine task
      Updating task transcripts . . .
Title:  Document DSEE learning procedure.
.
-----
Completed Items
.
x Replaced mgr_note[1] in library
  "//lush/libby/dsee_examples/project_doc"
As part of the task entitled:
Document DSEE learning procedure.
-----
Original document from manager
Completed:  4-Dec-1990 16:07
Completor:  Chris Libby on //LUSH (AD1)
DSEE> show history mgr_note
Replace mgr_note[1]
  4-Dec-1990 16:07  Chris Libby at //LUSH
    (chris.none.eng.AD1)
    As part of the task entitled:
    Document DSEE learning procedure.
    -----
    Original document from manager.
Create Element mgr_note
.
```

Figure 6-7. Automatic Cross-Referencing

Using Monitors

When you're working on a project with many other people, you can't always count on them to let you know when their work may affect you. What you *can* do is use the DSEE monitor mechanism to "watch" elements on which your own work depends. When a monitored element is renamed, deleted, or given a new version (by a `replace` command), the DSEE facility can let you (or anyone else) know of the event.

When a monitor is activated, it can execute one or more shell commands and/or add tasks to tasklists. The benefits of monitors that execute shell commands are great. You can have the DSEE facility do such things as send mail and copy files automatically whenever an element is modified. Monitors that create and add tasks to tasklists are also very useful, especially when you consider that a change to an element generally means that you should perform some task. For example, every time someone changes an include file that you use, you should probably investigate the change and evaluate its impact on your work. You can create a monitor that watches the include file and adds to your tasklist a task with the active items "Investigate change" and "Evaluate impact" whenever the include file was changed.

By default, the `alarm_server` program creates a new window and sounds a "beep" whenever a new task is added to your personal tasklist. If your personal startup file executes `alarm_server` every time you login, you will always see an alarm window whenever a monitor adds a new task to your personal tasklist.

NOTE: In order for a node to receive alarms from the `alarm_server` program, the Display Manager must own the background (root) window of the node.

Creating a Monitor

A monitor is associated with two libraries: the one that's current when the monitor is created, and the one containing the element or elements being monitored. The first library is said to own the monitor; the second library is said to hold the monitor.

Because monitors are associated with two libraries, you can monitor elements in libraries that belong to other development groups. If, for example, your system depends on an include file that is maintained by people in another project, you may want to have a monitor on the element containing the include file. You would create the monitor while one of your own project's libraries was set as the current library; this library would be the owner of the monitor. The library in which the include file resides would be the holder of the monitor. (It's not necessary, though, to have two different libraries. One library can both hold and own a monitor.)

You can create a monitor that watches one or more specific elements, or you can create a monitor that watches a group of elements whose names match a wildcard. When you create a monitor that watches elements whose names match a wildcard, the monitor will also watch any new elements added to the library whose names match the wildcard.

You create a monitor with the **create monitor** command. When you create a monitor, the DSEE facility asks you for several pieces of information:

- The purpose of the monitor
- The tasklists to which new tasks are to be added and/or the shell commands to be executed when the monitor is activated (known collectively as the **activation list**)
- The title of the task to be added to tasklists (if you put any tasklists in the activation list)

If your monitor will create a task, the DSEE facility next asks whether you want to edit the task. If you respond by selecting Yes, the DSEE facility invokes the DSEE task editor, and you edit the task as described earlier in this chapter. (This task is actually a template for the tasks that the monitor will create when it is activated. Each time the monitor is activated, the DSEE facility uses this template as the text of the new task that it adds to the specified tasklists.)

A monitor that watches the project documentation library you created earlier in this chapter would help you keep abreast of any changes to project procedures and protocol. You can create a monitor that watches all of the elements (the present ones and any

created in the future) in this new library by issuing the **create monitor** command as follows:

1. Select **create monitor** from the Monitor menu.
2. The DSEE facility displays a dialog box. By default, the buttons next to **monitor is associated with the current library** and **for anyone** are selected. In the top text entry box type

?*
3. Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box. This one asks you to enter the purpose of the monitor. Type

This monitor watches all the elements in the project documentation library and lets me and my manager know when one is added, changed, or deleted.
5. Point to the Confirm button and click <M1>.
6. The next dialog box asks you to enter an activation list. Figure 6-8 shows an example of this step in the monitor creation process. As shown in the figure, the DSEE facility helps you out by telling you (in the transcript area) the format of entries in the activation list.

In the example, the monitor being created will do two things when activated: it will create a task (using the template defined a little later in the monitor creation process) and add it to the tasklist `^libby`, and it will copy the most recent version of the element being added, changed, or deleted to the directory `//lush/libby`.

Notice the ``dsee_element` in the dialog box in the figure. This is called an activation string. When someone activates the monitor, the DSEE facility substitutes the accurate value for any activation strings in the monitor. In this case, the DSEE facility substitutes the name of the element being created, revised, or deleted when the monitor is activated. (Other activation strings are explained in the description of the **create monitor** command in the *DSEE Reference*.)

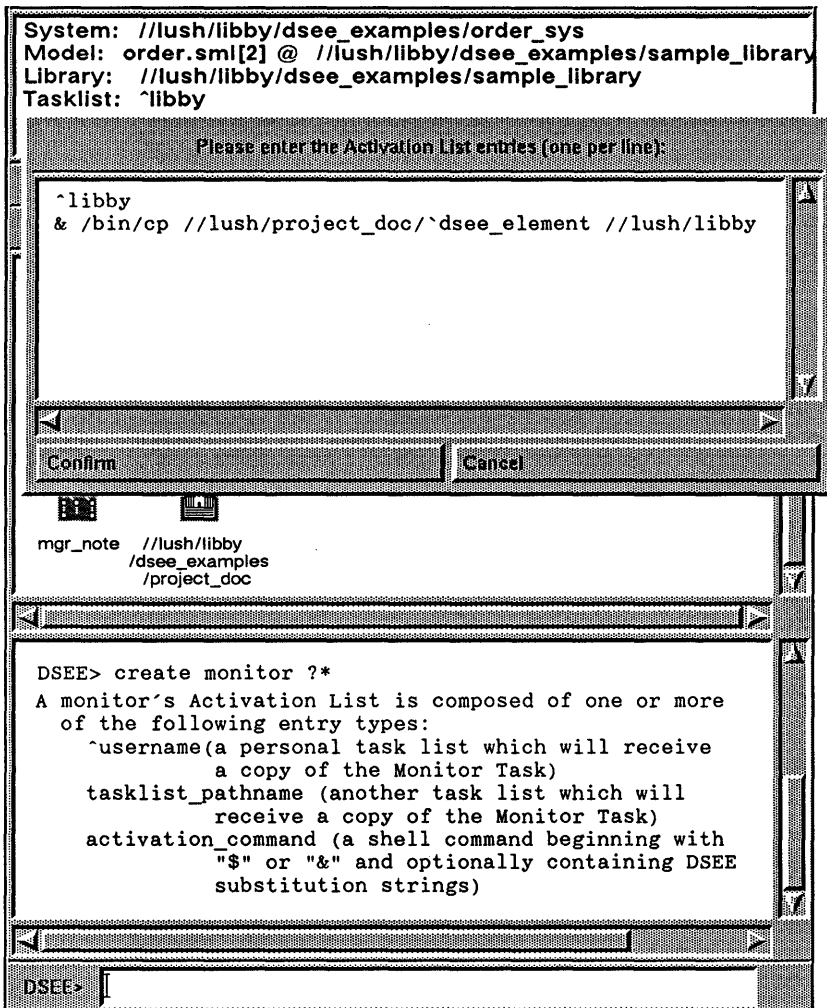


Figure 6-8. A Monitor's Activation List

Modify the activation list items shown in Figure 6-8 for your own system. Type in the name of your own personal tasklist (^userid). You can also enter a UNIX shell /bin/cp command like the one in the example, or an Aegis /com/cpf command to copy the most recent version of the element into your home directory. Give the full pathname of the project_doc library as the first argument to the cp

command, and give the pathname of your home directory as the second argument. Precede the `cp` command with an ampersand, as in the example. (The ampersand (&) causes the `cp` command to be executed as a background process rather than as a foreground process.)

7. When you've finished your activation list, point to the Confirm button and click <M1>.
8. The next dialog box asks you to supply a title for the task that will be added to your personal tasklist when the monitor is activated. Type

Investigate change to element
9. Point to the Confirm button and click <M1>.
10. Finally, the DSEE facility displays a query box that asks you whether you want to edit the new task. Selecting Yes invokes the task editor; you can enter anything in the task that you want. However, you needn't edit the task at all. In that case, the new task added to your tasklist each time the monitor is activated will contain only a title.

Activating a Monitor

There are two aspects to activating a monitor: what the person activating the monitor sees, and what operations the monitor performs when it is activated. Because you will be activating a monitor that adds a task to your personal tasklist, you will see both aspects of the process.

Any DSEE command that causes a change in an element (that is, any command that creates a new element version, renames an element, or deletes an element) can activate a monitor. To make sure that you know about a monitor *before* you create a new version of an element, commands that activate monitors tell you if you will activate a monitor. In addition, commands that prepare you to create new versions of elements (such as `reserve`, which would be followed by `replace`) tell you about any monitors on target elements.

If you want an automatic alarm window to appear every time a new task is added to your personal tasklist, make sure that your personal startup file contains a `cpo` command to execute the `alarm_server`. The command can have one or more options; however, at the minimum it should look like this:

```
cpo /sys/alarm/alarm_server
```

NOTE: In order for a node to receive alarms from the `alarm_server` program, the Display Manager must own the background (root) window of the node.

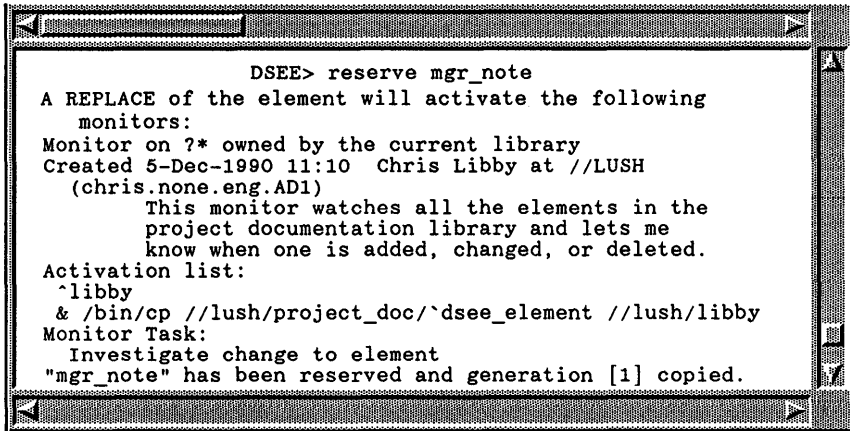
Once `alarm_server` has been notified about a new task on your personal tasklist, it will continue to notify you of that new task by popping up an alarm window each time you login until you issue the `examine tlist` command and examine your personal tasklist.

To activate your monitor, reserve the element `mgr_note` in order to add some information about the new library you created. Figure 6-9 illustrates the DSEE responses to the `reserve` command.

1. Point to the branch icon for `mgr_note` and click <M1>.
2. Select `reserve` from the Branch menu.
3. The DSEE facility displays a dialog box seeded with `mgr_note`. Point to the Confirm button and click <M1>.
4. The DSEE facility displays another dialog box. This one asks you to comment on the reservation. Append the following comment to the text in the dialog box:

Adding library information

5. Point to the Confirm button and click <M1>.



```
DSEE> reserve mgr_note
A REPLACE of the element will activate the following
monitors:
Monitor on ?* owned by the current library
Created 5-Dec-1990 11:10 Chris Libby at //LUSH
(chris.none.eng.AD1)
This monitor watches all the elements in the
project documentation library and lets me
know when one is added, changed, or deleted.
Activation list:
^libby
& /bin/cp //lush/project_doc/`dsee_element //lush/libby
Monitor Task:
Investigate change to element
"mgr_note" has been reserved and generation [1] copied.
```

Figure 6-9. Activating a Monitor

Edit the copy of the element in your working directory to reflect the creation of the new library for project procedure documents.

Replace the element **mgr_note**.

1. Select **replace** from the Branch menu.
2. The DSEE facility displays a dialog box seeded with **mgr_note**. Select the **-nc** option.
3. Point to the Confirm button and click <M1>.

When you replace the element, the DSEE facility tells you (in the transcript area) that it is activating the monitor.

Some time soon after you've replaced the element, **alarm_server** will place a window on your screen. If you are running the Display Manager, the window will look like the one shown in Figure 6-10.

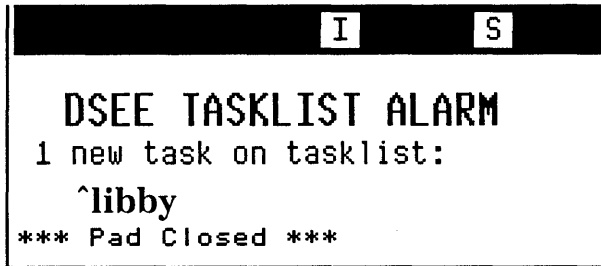


Figure 6-10. An Alarm Window

You can scroll this window to view its full contents, which include information about the operation that activated the monitor. You can also see this same information by examining your tasklist, as illustrated in Figure 6-11.

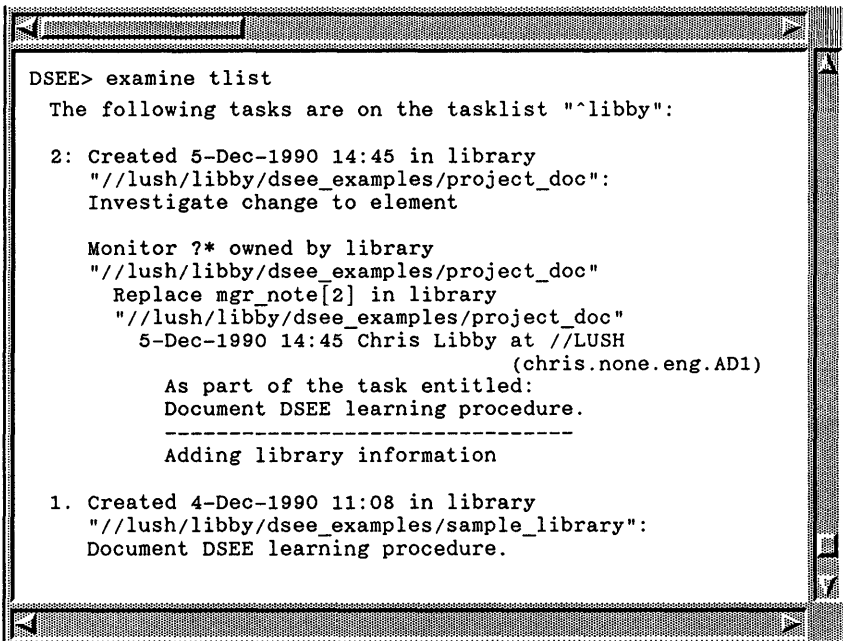


Figure 6-11. A Tasklist with a New Task

Deleting a Monitor

If you determine that a monitor is no longer useful, you can delete it. The **delete monitor** command deletes an existing monitor.

In order to identify the monitor you want to delete, you must use exactly the same wildcard or element name that you used when you created the monitor.

1. Select **delete monitor** from the Monitor menu.
2. The DSEE facility displays a dialog box. In the top text entry box, type

?*

This is the same wildcard you used when you issued the **create monitor** command.

3. Point to the Confirm button and click <M1>.
4. The transcript area displays the creation comments, activation list, and (if applicable) the title of the monitor task. Also, the DSEE facility displays a query box that asks if you want to delete the monitor. Point to the Yes button and click <M1>.

Command Summary

This chapter presents only part of the DSEE project management facility. Here are the commands that we presented in this chapter. You can obtain more information about them from the *DSEE Reference*.

Monitor

create monitor

Creates a monitor and sets it on one or more elements.

delete monitor

Deletes a monitor previously set on one or more elements.

Task

- create task** Creates a new task.
- edit task** Invokes the DSEE task editor for editing a task.
- examine task** Displays the current state of a task.

Tasklist

- examine tlist** Displays the current state of a tasklist.
- set tlist** Sets or clears the current tasklist setting.

Related Information

There are many other DSEE commands that deal with tasks, tasklists, and monitors. Brief descriptions follow. More information is available in the *DSEE Reference*.

Form

- create form** Creates a form.
- delete form** Deletes a form.
- edit form** Invokes the DSEE task editor to edit a form.
- examine form** Displays the current state of a form.

Monitor

edit monitor Invokes the DSEE task editor to edit one or more monitor task templates.

recover monitor

Deletes a monitor held by the current library when it cannot be deleted normally with the **delete monitor** command.

show monitors

If invoked with the **held** option, displays monitors that are currently set to watch elements in the current library. If invoked with the **owned** option, displays monitors that were created in the current library.

Task

add task Adds a task reference to a tasklist.

delete task Deletes a task from a tasklist.

set task Sets or clears the current task setting.

show task Displays information about the current task.

tag task Tags a version of an element with the current task, or with no task.

Tasklist

create tlist Creates a new tasklist.

delete tlist Deletes a tasklist.

protect tlist Sets or shows protection attributes for a tasklist.

recover tlist Updates the task references on a tasklist.

show tlist Displays the name of the current tasklist.
watch tlist Specifies tasklists to be watched by **alarm_server**.



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

Glossary

access rights

A user's authority to examine and/or modify DSEE libraries and tasklists. *See* protection attributes.

activation command string

A generic argument that can be used in an executable entry in a monitor's activation list. Activation command strings allow you to compose shell commands that manipulate DSEE objects, and to specify a particular object as a variable to be resolved at the time of execution.

activation list

A list of the tasklists and/or executable shell commands associated with a monitor. *See also* task.

active item

A "to-do" step within a task.

active task

A task that appears on one or more personal or library tasklists. An active task usually has at least one active item.

administrator

The highest user class in the DSEE protection hierarchy. An administrator has full access to DSEE objects. Administrators can view a library, modify and delete its contents, delete the library, and change its protection. Also refers to a person who assumes the tasks of maintaining DSEE systems. Typically, an administrator creates and maintains libraries and system models, and may also perform other tasks to coordinate DSEE projects.

age limit

The amount of time, in hours, that a particular set of build results must sit in a pool before it becomes a candidate for deletion. The **create pool** and **configure pool** commands allow you to define this parameter.

aggregate block

A system model block that does not represent any source component. Typically, an aggregate block contains subordinate blocks. An aggregate block starts with the keyword **aggregate**.

alarm_server

A Domain/OS system server process that monitors a variety of system conditions, including additions to DSEE tasklists. In response to one of these conditions, **alarm_server** displays an alarm window.

alarm window

A window, created by **alarm_server**, that displays information about the event that triggered **alarm_server**.

alias

In a system model, a substitute name for a text string. The **alias** declaration defines aliases.

alternate line of descent

See branch.

base version

An element version that is the youngest common ancestor of two versions being merged into a line of descent. The DSEE facility uses the base version to help resolve differences between the two objects being merged.

Bcomp

See buildable component.

BCT

See bound configuration thread.

binary pool

See pool.

block

A group of system model declarations that describes an element, external, aggregate, or model. The hierarchically arranged units of a system model. A block begins with the keyword **element**, **external**, **aggregate**, or **model**.

block ID

The name by which blocks refer to another block. The name of a block.

bound configuration thread

An output of a build that describes the sources that were used to perform the build. A bound configuration thread states which element versions and which translator options were used to build a component. The DSEE facility produces a bound configuration thread for each component it builds. Also called a BCT.

branch

A line of descent other than an element's main line of descent. Also called alternate line of descent.

branch name

The name you assign to a branch. You name a branch when you create it with the **create branch** command; you can change its name with the **rename branch** command.

branch path

The path to a branch, used as an argument in some DSEE commands. Much like a file's pathname, a branch path describes the path to a branch by listing all the branches that lead to that branch. Branch names in a branch path are separated by slashes. For example, **/br1/br2/br3** is the path to branch **br3**.

browse box

A dialog box displayed by the **Browse** command which presents a list of objects that you can display as icons on the desktop.

build

To translate a component according to the translation rules specified in the system model and using the versions and options specified in the configuration thread. A build always produces a bound configuration thread; it may produce one or more derived objects, as well.

build

The act of building a component. ("I'm about to initiate a build.") Also, an occurrence of building. ("What were the results of that build?")

build map

A user-readable form of a BCT. The **show builds** command displays build map headers.

build name

See full build name.

build process

The process in which a build begins executing. This process may reside on the command node or on a helper node.

build results

The BCT and derived objects, if any, produced by a build.

build specifier

In a configuration thread, a descriptor that identifies a specific build.

build unit

See buildable component.

build unit specifier

The ID of a system model block representing a build unit.

build version

The date and time of a build as displayed by the **show builds** command.

buildable component

An element, aggregate, external, or model that's defined in a system model and has a translation rule (defined by the keywords **translate** or **nil_translation**). Anything that can legally serve as an argument to the build command. Also called Bcomp and build unit.

builder

A computer on which a translation executes.

builder list

A file named **.dsee_builder_list**, located in your working directory, your home directory, your **~/user_data** directory, or the **/sys/dsee/dsee_config** directory. The builder list identifies the candidate builders and other information for each host type. Also, a list of candidate builders specified by the **set builders** command.

Cancel button

A DSEE desktop button which closes a dialog box without executing a command.

command node

The computer on which the DSEE software is running.

comment

A user-supplied text string describing an object (such as a library) or event (such as a build). By default, the DSEE facility asks you for a comment whenever you create or change a DSEE object. Comments become part of the history that the DSEE facility records for each object.

completed item

See transcript item.

component

An element, aggregate, external, or model defined by a block in a system model; any element, file, or tool referenced as a dependency in a system model.

configuration manager

The part of the DSEE facility that allows you to build and maintain versions of entire systems. The configuration manager enables different users to build systems simultaneously, share common object code, and rebuild systems.

configuration thread

A list of rules that identify the element versions and translator options to be used during builds.

Confirm button

A DSEE desktop button which accepts the information in a dialog box and proceeds with the command.

context

Within a system model, the objects visible from a particular block.

context banner

A banner at the top of the DSEE desktop interface or Display Manager interface that indicates a current setting, such as the current library, system model, or directory.

current configuration thread

The configuration thread associated at present with the current system. When you build a component, the DSEE facility uses the current configuration thread to determine which element versions and which translator options to use. The **set thread** command sets a current configuration thread.

current library

The library in which a user is currently working. The **set library** command sets a current library.

current model thread

The model thread associated at present with the current system. When you set a current system model, the DSEE facility uses the current model thread to determine which versions of model fragments to use in composing the system model. The **set thread** command sets a current model. *See also* model thread.

current system

The system directory that contains internal information about the system on which a user is currently working. The **set system** command sets the current system.

current system model

The system model associated at present with the current system. You can build a component that is defined in the current system. The **set model** command sets the current system model.

current task

The task on which a user is currently working. When you replace a line of descent, the DSEE facility records that activity in the current task. The **set task** command sets the current task.

current tasklist

The tasklist from which a user is currently working. The **set tlist** command sets the current tasklist.

default declaration

A system model declaration that pertains to multiple components. Default declarations are identified by the keyword **default**.

dependency

A source file, build result, or translator required for building a system model component. *See* direct dependency, primary source dependency, result dependency, source dependency, tools dependency.

derived object

An object produced by a translation. Part of a build result.

desired BCT

The BCT required to build a particular component. The DSEE facility determines a component's desired BCT based on the component's description in the system model and the version information in the current configuration thread.

desktop

A DSEE interface based on the X Window System and OSF Motif. The desktop is the default DSEE interface.

dialog box

A window displayed by the DSEE desktop in which users select command options and enter command arguments. The DSEE desktop displays a dialog box in response to the selection of a command.

direct dependency

The items explicitly listed as a component's dependencies: those listed in its **depends_source**, **depends_tools**, and/or **depends_result** declarations, and/or its primary source dependency. The DSEE facility uses direct dependencies to

determine whether to build a component. *See* result dependency, source dependency, tools dependency.

Dismiss button

A DSEE desktop button which closes windows that simply display text.

DSEE

Domain Software Engineering Environment.

element

A user-defined, versioned file that resides in a DSEE library.

element block

A system model block, identified by the **element** keyword, that represents an element in a DSEE library. The element is known as the block's primary source dependency.

element change record

A notice of a change to a library element. The DSEE facility places element change records in the transcript of the current task. An element change record notes the operation performed on the element, who performed it, and when. Any comments written by the user describing the operation are also included.

element version

See version.

equivalence

An existing derived object that is interchangeable with the not-yet-created derived object described by the desired BCT. You can declare equivalences when you issue the **build** command.

export

To copy derived objects from a pool to a stable directory.

external

A system model language declaration that defines an external block.

external block

A system model block, identified by the **external** keyword, that represents an ordinary file (not a DSEE element). The file is known as the block's primary source dependency.

filename

The name of a file, without any directory specification. For example, the filename of the file `//philadelphia/joe/myfile` is `myfile`. *Also* leafname.

form

A user-defined description of a standard procedure. A form can be used as a template for a task with the `-from` option to the **create task** command.

full build name

The name of an existing build. The name consists of two parts, separated by an exclamation point (!). The left side of the exclamation point identifies the component that was built. The right side identifies either the date and time when the build was performed, or the release directory in which the build is stored. The **show builds** command displays the full build names of all builds whose results are currently stored in pools.

history

The element change records and comments associated with a library, its elements, and their branches. The **show history** command displays the history of a library or of one or more elements.

hold

To contain the elements watched by a monitor. ("The `color_lib` library holds the monitor that watches `blue`, `green`, and `red`.")

host type

A class of computer on which a buildable component is to be built. By default, this host type is `apollo`, but you can specify, in your system model, other host types for buildable components. Host types are defined in the file, `/sys/dsee/dsee_config/hosts`.

icon

A graphic symbol that represents a DSEE object. The DSEE desktop interface can display icons to represent individual libraries, elements, versions, lines of descent, buildable components, build results, files, pools, and translators.

icon area

An area of the DSEE desktop that displays icons for different kinds of objects, such as libraries and elements.

leafname

The name of a file or element, without any directory or library specification. For example, the leafname of the file `//philadelphia/joe/myfile` is `myfile`. Also called filename.

library

A directory that you instruct the DSEE facility to create. A library contains elements, tasks, and tasklists.

limit

The maximum number of build results of a single component that a pool can contain before the least recently used versions become candidates for deletion. The **create pool** and **configure pool** commands allow you to define this parameter.

line of descent

A path of development of an element. Every element has a main line of descent. You can create alternate lines of descent, known as branches.

<M1>

The leftmost mouse button.

<M2>

The middle mouse button.

<M3>

The rightmost mouse button.

main line of descent

The central path of development of an element. It begins with the initial version of the element and continues up through the most recent version of the element. It does not include any branches that may exist off the main line of descent.

master tasklist

A tasklist that references all current and deleted tasks in a library. The DSEE facility creates a master tasklist for a library when it creates the library. A master tasklist's name is the library's pathname followed by `-master`.

member

Second highest ranking user class in the DSEE protection hierarchy. In general, a member can view and modify elements, tasks, tasklists, and monitors.

merge

To interleave the text of two versions, the text of a version and another file, or the text of two files. The result is a file called the merge file. The `merge -fetch` command performs this type of merge. Also, to interleave the text of a version on one line of descent with the most recent version on another line of descent (of the same element). The result is a file called the merge file, and the reservation of the second line of descent. Replacing the line of descent makes the merger file the latest version on that line of descent. The `merge -reserve` command performs this type of merge.

Also, the act of merging.

merge file

The file created by a merge.

model thread

A list of rules that identify the versions of model fragments to use for the system model, and the conditional processing variables to define for the system model. *See also* current model thread.

monitor

A flag set on an element that directs the DSEE facility to watch that element and add a task to a tasklist and/or execute shell commands when that element is changed. The **create monitor** command sets the monitor flag on one or more elements.

noncritical translator option

An option to a translation rule that does not become a key attribute of the resulting derived object. The presence or absence of noncritical options does not affect whether the DSEE facility considers an existing derived object to be a suitable match for the desired derived object.

non-user

The lowest ranking user class in the DSEE protection hierarchy. A non-user has no access to DSEE objects. A person is a non-user if his or her Subject IDentifier (SID) is omitted from a library or tasklist's protection attributes or explicitly paired with the non-user class (an explicit non-user).

obsolete

Applied to a line of descent, closed to the addition of any new versions. The **obsolete** command makes a line of descent obsolete. The **cancel obsolete** command reactivates an obsolete line of descent.

override

An identifier used with the **build** command that specifies that an existing derived object is interchangeable with the not-yet-created derived object described by the desired BCT. An override differs from an equivalence in that the override is in effect only during a specific build. An equivalence is in effect indefinitely.

own

To have been the current library when a monitor was created. ("The **color_lib** library holds the monitor that watches **blue**, **green**, and **red**, but the **draw_lib** library owns the monitor.")

personal tasklist

A tasklist that the DSEE facility creates for every new user in the `user_data` subdirectory of the user's home directory. The name of your personal tasklist is your account name preceded by a caret in the form `^name`. This tasklist generally contains references to all of your assigned tasks.

pool

A storage area for derived objects and BCTs. The DSEE facility automatically creates a default pool for each system. You can create alternate pools with the `create pool` command.

pop-up menu

A menu selected by pointing the cursor in the icon area of the DSEE desktop and pressing or clicking `<M3>`.

primary source dependency

The element, file, or model represented by an element block, external block, or model block in a system model. (Aggregate blocks don't have primary source dependencies.) A block's primary source dependency is considered one of its direct source dependencies, even though it isn't explicitly listed in the block's `depends_source` statement.

promote

To make the direct dependencies of one system model component become direct dependencies of any other component that references the first component. The system model `promote_depends` declaration instructs the DSEE facility to promote a component's dependencies. Also, to move a derived object from a reserved pool to a user-defined pool or to a default pool. The DSEE facility automatically promotes a component's derived objects when all of the component's element dependencies are replaced. You can also use the `promote` command to promote derived objects explicitly.

protection attributes

DSEE protection rights for libraries and tasklists. Each protection attribute consists of a DSEE user class (administrator, member, reader, non-user) and a corresponding Subject IDentifier (SID).

pull-down menu

A menu selected by pointing the cursor at a menu title in the menu bar of the DSEE desktop and pressing or clicking <M1>.

reader

Third highest ranking user class in the DSEE protection hierarchy. A reader can read elements and the contents of tasklists, but cannot modify them.

reference node

The computer intended as the single reference point for all relative pathnames interpreted by a single host type. *See* reference path.

reference path

The directory, on the reference node, intended as the single reference point for all relative pathnames interpreted by a single host type. You define reference paths for each host type in the builder list.

release

To copy derived objects and BCTs from pools to release areas. The **create release** and **edit release** commands release build results.

Also, a set of build results that have been released.

release area

A tree of directories constructed by the DSEE facility to hold releases. A release area is referred to by the name of the release directory that you create with a **create release** command.

release directory pathname

The name of a release directory, which you define using the **create release** command.

replace

To un-reserve a line of descent. When you replace a line of descent, the DSEE facility creates a new version on that line of descent. As the contents of the new version, the DSEE facility uses the contents of a file in your current working directory with the same name as the line of descent's element. (When you reserved the line of descent, the DSEE facility created that file in your working directory. You may have revised it before replacing the line of descent.)

reserve

To express your intent to create a new version on a line of descent. When you reserve a line of descent, the DSEE facility prevents anyone else from reserving it, and creates a copy of its most recent version in your working directory. (You can modify the copy, then add it to the line of descent as a new version by issuing the **replace** command.)

reserved pool

A pool created by the DSEE facility to store derived objects that used reserved versions (that is, derived objects that used the copies created by the DSEE facility when you reserved lines of descent).

result dependency

A component whose build results are required in order to build another component. The system model **depends_result** declaration identifies result dependencies. ("The component **book** has a result dependency on the component **chapter**. The component **chapter** is a result dependency of the component **book**.")

rule

A statement in a configuration thread that identifies which element versions to use in a build and which translation options to use in a build. Also, a statement in a model thread that identifies the versions of model fragments to use for the system model, and the conditional processing variables to define for the system model.

scroll bar

The part of the DSEE desktop that enables you to view more text than will fit in the transcript area at a time, or view icons that are outside the viewing area of the icon area.

seed

To insert information into an edit pad or a dialog box, based on information previously supplied by the user.

server process

An auxiliary process that performs a supporting service for one or more other pieces of software. The DSEE facility requires that these server processes run on command nodes: **d3m_server**, **mbx_helper**, and **sf_helper**. The DSEE facility can make use of the optional server process **alarm_server**. Builder nodes must run the **server_process_manager**.

source dependency

A source file required to build a component. Each component has a source dependency on the element, external, or model that it represents; this dependency is known as the primary source dependency. All other source dependencies are listed in the component's **depends_source** declaration.

source reference environment

Within a process, the default versions of all elements, that is, the versions that the DSEE facility will access if you specify element names without specifying any branches or versions. The default source reference environment is the most recent version on each element's main line of descent. You can change the source reference environment in a process by issuing the set environment command. You can create a new process with a specific source reference environment by issuing the create environment command.

NOTE: The source reference environment does *not* affect the versions used during a build; those versions are determined by the configuration thread.

source version

A version that is being merged into another line of descent of the same element.

system

A collection of interacting or interrelated software modules. *See also* system directory.

system component

See component.

system directory

A subdirectory that the DSEE facility creates, at your request, to store the internal information and objects required to build a system. A system directory also can contain references to release areas that hold released builds of the system's components.

system model

A user-defined specification for a system build that lists the components of the system, their dependency relationships, and the translation rules (such as compiler, binder, and formatter command lines) required to build them. A system model can be an ordinary Domain/OS file or a DSEE element.

target line of descent

The line of descent on which the DSEE facility creates a new version as a result of a **merge -reserve** command and a subsequent **replace** command.

task

A definable job or chore. The work of a programming project can be expressed in the DSEE facility in terms of the tasks involved. A DSEE task consists of a header (title and creation information), a user-defined list of the low-level steps required to complete the high-level activity (its **active items**), and a transcript consisting of completed items and element change records relating to the high-level activity.

task editor

A graphics-based program that enables users to modify the contents of a task, monitor template, or a form through menu choices.

task number

An integer that identifies a task's relative position on a tasklist. The **examine tlist** command displays tasks with their numbers.

task template

A prototype, or model, for a task. A task template consists of a user-supplied title, creation information, and, optionally, active items. When you activate a monitor, the DSEE facility uses the monitor's task templates to instantiate (create editable copies of) actual tasks.

task transcript

The portion of a task containing completed items and element change records.

tasklist

A list of tasks maintained for a given user (personal tasklist), for a given library (active tasklist and master tasklist), or for the project in general (a nonlibrary tasklist).

thread

See configuration thread and model thread.

tools dependency

A translator, shell script, or other tool used in the translation rule of a system component. A component's tools dependencies are listed in the system model in a **depends_tools** declaration.

transcript

To record an operation on an element in the transcript area of the current task.

transcript area

The part of the DSEE desktop that the DSEE facility uses to echo the commands you issue, to display messages, and, where appropriate, to display the results of commands.

transcript item

A task item that you've moved from the task's active list to its completed list (typically after completing the item). The DSEE facility also records **replace** operations as completed items in the current task.

translation rule

The specification of the sequence of shell commands required to bind, compile, assemble, or otherwise process a system component. A translation rule is declared in a system model **translate** declaration. Each translation rule includes all the required and potential options to perform the translation.

user class

A DSEE user's access classification. The DSEE facility defines four user classes: administrator, member, reader, and non-user. A user's class determines to what extent he or she can access DSEE libraries and tasklists.

version

An instance of an element. Each time you reserve and then replace an element, the DSEE facility creates a new version to represent the new instance of the element. Older instances of the element still exist as previous versions. You can access a version by specifying the element name followed by the version number or version name. For example, you can access version 20 of element **beta.pas** by specifying **beta.pas[20]**, or version "rev01" of element **gamma.pas** by specifying **gamma.pas[rev01]**.

version

Also, to maintain separate instances of. ("The DSEE facility versions elements.") Also, to retrieve appropriate versions of. ("The DSEE facility versions elements that are declared as source dependencies.")

version name

A name assigned with the **name version** command that identifies an element version. Users typically choose version names that suggest a common characteristic of a group of element versions. For example, you might assign the version name **sr5** to the element versions that constitute Software Release 5.0.

version number

An integer in square brackets, often appended to an element name. A version number identifies a version by identifying which instance of the element it is.

version specifier

The representation of a specific version of a DSEE element. A version specifier can be a version number, a branch path followed by a version number, or a version name (enclosed in square brackets).

working context

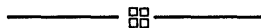
A working environment within a DSEE session. Your working context consists of one or more of the following parameters: current library, current task, current tasklist, current system, current system model, current configuration thread, current model thread, current source reference environment, and current shell. The DSEE facility remembers and restores your last working context when you enter the DSEE facility and changes your working context when you issue the **set system** command. You can save and reuse working contexts with the **set system** command's **-into** and **-from** options.

X resources

A set of customizable features for an application that you can specify using X Window System procedures.

X Window System

A network-based graphical windowing system developed by MIT.



Index

Symbols are listed at the beginning of the index.

Symbols

[] (square brackets), 3-12
% (percent sign), 3-6, 4-32
/ (slash), 2-23

A

abbreviating, blocks, 3-7
aborting commands, 3-8
activation list, 6-19
 See also monitors, activation list
activation string. *See* monitors, activation string
active items, 6-2
add task command, 6-26
administrator of a library, 2-3
age parameter, 3-20
aggregate blocks, 3-5
aggregate keyword, 3-6

alarm_server, 6-16, 6-21
 for non-personal tasklists, 6-27
 stopping automatic alarm windows, 6-21
alias declaration, 3-8
aliases in system model, 3-8
alternate lines of descent, 2-22

B

base version, 2-25
BCT, 3-1
-bct_only option (build command), 4-24 to 4-25
BCTs
 as sole product of build, 4-24 to 4-25
 BCT lookup during builds, 3-19
 comparing, 4-26
 copied to a release directory, 5-4

- definition of, 3-18
 - desired BCT, 3-19
 - examining, 3-20
 - in reserved pools, 4-10
 - storing, 5-1 to 5-6
- binary pools
- accessing contents of, 3-23 to 3-26
 - changing parameters of, 4-38
 - cleaning up, 4-38
 - creating, 4-38
 - default binary pool, 3-18
 - definition of, 3-18
 - deleting, 4-38
 - displaying information about, 4-38
 - lifespan of contents, 3-23
 - lifespan of objects within, 3-20
 - parameters, 3-23
 - parameters of, 3-20
 - promoting derived objects to, 4-37
 - promoting from reserved pools to, 4-8, 4-16 to 4-17
 - recovering after system failure, 4-38
 - types
 - default, 3-18
 - reserved, 4-8
 - reserved pool, 4-29
- block. *See* system model, block
- bound configuration thread. *See* BCT
- branches
- creating, 2-22 to 2-24
 - deactivating, 2-38
 - deleting, 2-37
 - displaying icons for, 2-16
 - displaying leaf names, 2-39
 - merging, 2-25 to 2-32
 - names, 2-23
 - reactivating, 2-37
 - renaming, 2-38
 - replacing versions on, 2-23
 - syntax for referring to, 2-23
 - why they are useful, 2-22
- browse box, 2-7
- Browse** command, 2-36
- browsing a library, 2-6
 - browsing for build units, 3-23, 3-27
 - browsing for builds, 3-23, 3-27, 4-18, 4-25, 4-36
 - browsing the system model, 3-23, 3-27
 - displaying elements, 2-6
 - listing branches with a particular name, 2-26
- build** command, 3-15 to 3-18, 3-26, 4-7, 4-36
- bct_only** option, 4-24 to 4-25, 4-36
 - query** option, 4-22 to 4-23, 4-27 to 4-31, 4-36
- build map, 3-20
- build maps
- comparing, 4-26
 - header, 3-20
 - reading, 5-5
 - storing, 5-2, 5-4
- build option specifications, 4-32
- build servers, 4-38
- build unit specifier, 4-15
- definition of, 3-17
- build units
- browsing the system model for, 3-23, 3-27
 - definition of, 3-17
 - displaying an icon for, 3-24

- build version, 3-20, 3-21
- build versions, definition of, 3-17
- builder list, 4-12
- builds, 3-1 to 3-28, 4-1 to 4-38
 - BCT only, 4-24 to 4-25
 - browsing for, 4-18, 4-25
 - build process, 3-18 to 3-20
 - build versions, 3-17
 - building systems, 3-15 to 3-18
 - comparing, 4-25 to 4-26
 - concurrent builds, 4-11 to 4-14
 - deleting, 4-37
 - determining what needs to be built, 4-22 to 4-23
 - displaying an icon for, 3-24, 4-18, 4-25
 - examining, 3-20 to 3-23
 - executing built programs, 3-23
 - exporting, 4-15
 - for release, 4-31 to 4-36
 - full build names, 3-17
 - full build names using released builds, 5-5
 - how model and thread are used, 3-10, 3-19
 - how older builds are used, 3-19
 - identifying person who built, 3-20
 - identifying substitute builds, 4-27
 - initiating, 3-15 to 3-18
 - introduction, 1-4
 - map. *See* build maps
 - names (for most recent builds), 4-15
 - names after promotion, 4-17

- naming versions used in, 4-17 to 4-21
- obtaining information about concurrent, 4-38
- obtaining names of, 3-21
- overriding, 4-26
- parallel, 4-11 to 4-14
- partial builds, 4-7 to 4-11
- prerequisites, 3-2
- reconstructing older builds, 4-31 to 4-36
- referring to in configuration threads, 3-11
- reserved element builds, 4-8, 4-14
- reusing older builds, 4-14, 4-21 to 4-36
- simple builds, 3-15 to 3-18
- single component builds, 4-7 to 4-9
- specification, 3-19
- speeding up, 4-11 to 4-14
- storage area for. *See* binary pools
- storing, 3-23, 5-1 to 5-6
- storing results of, 3-18
- system rebuilds, 4-11 to 4-21
- unsuccessful builds, 4-9
- using versions in a shell, 4-37
- with reserved versions, 4-4
- without translation options, 4-32

C

- callable interface (DSEE), 1-18
- cancel obsolete command, 2-37
- cancel reserve command, 2-37

- capabilities of DSEE
 - environment, 1-2
- cd command, 2-3, 2-36
- classes of DSEE users, 2-3
- Clear Transcript** command, 1-15
- command files, 4-3
- comments, suppressing requests for, 2-23
- compare builds** command, 4-25
 - to 4-26, 4-36, 5-5
- compare** command, 2-19, 2-37
 - serial option, 2-19
- components
 - definition of, 3-4
 - dependencies, 3-4
 - selecting for rebuilding, 4-27 to 4-28
- concurrent building, obtaining information about, 4-38
- concurrent builds, 4-11 to 4-14
- configuration threads, 1-18, 3-10 to 3-14
 - build option rules, 3-11, 3-13
 - build-ID-based rules, 3-11
 - clearing current setting, 3-11
 - default thread, 3-11
 - definition, 1-4
 - editing, 3-13 to 3-14, 4-32 to 4-34
 - examining, 3-13
 - examining current, 4-6
 - for reconstructing older builds, 4-32 to 4-34
 - how used during builds, 3-19
 - how used in builds, 3-11
 - language, 3-10
 - in equivalence panes, 4-29
 - referring to released builds, 5-5
 - rule qualifiers, 3-10, 3-13
 - rules, 3-10
 - setting current, 3-11
 - syntax of rules, 3-10
 - using instead of equivalences, 4-27
 - using named versions, 4-33
 - using stored threads, 3-11
 - validating, 3-11
 - version rules, 3-10
 - when_exists flag, 4-33
- configure pool** command, 4-38
- context banner, 1-8, 2-5, 2-6, 2-12, 3-3
- context banners in the DSEE window, 6-4, 6-9
- contexts, setting, 2-3 to 2-6
- Copy Icon Name** command, 4-26, 4-36
- copying and pasting text, 1-8, 1-12, 3-21
- create branch** command, 2-22 to 2-39
- create element** command, 2-20 to 2-39
- create environment** command, 4-37
- create form** command, 6-25
- create library** command, 2-11 to 2-39
- create monitor** command, 6-18, 6-24
- create pool** command, 4-38
- create release** command, 5-1 to 5-5
- create system** command, 4-38

- create task command, 6-2 to 6-27
- create tlist command, 6-3, 6-26
- current
 - configuration thread, 3-12
 - library, 2-3
 - settings, 1-8
 - system, 3-2
 - system model, 3-8
- Cut Icon command, 2-34, 2-35

D

- .dsee_builder_list file, 3-16, 4-11
- declarations
 - aggregate, 3-6
 - alias, 3-8
 - default, 3-7
 - overriding, 4-6
 - definition of, 3-6
 - depends_result, 3-7
 - depends_source, 3-7
 - depends_tools, 3-7
 - element, 3-6
 - element (abbreviated form), 4-6
 - environment, 3-8
 - external, 3-6
 - external (abbreviated form), 3-7
 - host_type, 3-8
 - library, 3-8, 4-6
 - model, 3-6
 - shell, 3-8
 - system, 3-8
 - title, 3-8
 - translate, 3-6

- default
 - binary pool, 3-18
 - configuration thread, 3-11
- default declaration, 3-7
 - overriding, 4-6
 - qualified, 3-7
 - unqualified, 3-7
- delete branch command, 2-37
- delete build command, 4-37
- delete element command, 2-34, 2-35
- delete form command, 6-25
- delete library command, 2-38
- delete monitor command, 6-24
- delete pool command, 4-38
- delete release command, 5-6
- delete system command, 4-38
- delete task command, 6-26
- delete tlist command, 6-26
- delete version command, 2-39
- deleting text, 4-28
- dependencies, 3-4
 - component, 3-5
 - direct, 3-7
 - primary source, 3-6
 - primary source dependency, 4-1
 - result, 3-7
 - source, 3-7
 - source dependencies, 4-1
 - tool, 3-7
- depends_result declaration, 3-7
- depends_source declaration, 3-7

- depends_tools** declaration, 3-7
- derived objects
 - accessing, 3-23 to 3-26
 - copying to release directory, 5-4
 - creating links to, 3-23
 - definition of, 3-18
 - in reserved pools, 4-10
 - lifespan, 3-23
 - lifespan of, 3-20
 - names assigned to, 3-23
 - of reserved elements, 4-8, 4-14, 4-16 to 4-17
 - promoting from reserved pool, 4-37
 - reusing, 4-27 to 4-31
 - storing, 5-1 to 5-6
- desired BCT, 3-19
- desktop interface
 - context banner, 1-8
 - position indicator, 1-12
 - scroll bars, 1-10
 - using, 1-8 to 1-15
- direct dependencies, 3-7
- displaying
 - branch icons, 2-16
 - build icons, 3-25, 4-18, 4-25
 - element icons, 2-6
 - icons for build units, 3-24
- documentation (DSEE), 1-17 to 1-18
- %done** keyword, 3-6
- DSEE callable interface, 1-18
- dsee command, 1-7
- DSEE commands
 - add task, 6-26
 - Browse, 2-6 to 2-9, 2-26, 2-36, 3-23
 - build, 3-15 to 3-18, 3-26, 4-7, 4-36
 - bct_only option, 4-24 to 4-25
 - query option, 4-23, 4-27 to 4-31
 - cancel obsolete, 2-37
 - cancel reserve, 2-37
 - cd, 2-3, 2-36
 - compare, 2-19, 2-37
 - compare builds, 4-25 to 4-26, 4-36, 5-5
 - configure pool, 4-38
 - Copy Icon Name, 4-26, 4-36
 - create branch, 2-22 to 2-39
 - create element, 2-20 to 2-39
 - create environment, 4-37
 - create form, 6-25
 - create library, 2-11 to 2-39
 - create monitor, 6-18, 6-24
 - create pool, 4-38
 - create release, 5-1 to 5-5
 - create system, 4-38
 - create task, 6-2 to 6-27
 - create tlist, 6-3, 6-26
 - Cut Icon, 2-34, 2-35
 - delete branch, 2-37
 - delete build, 4-37
 - delete element, 2-34, 2-35
 - delete form, 6-25
 - delete library, 2-38
 - delete monitor, 6-24
 - delete pool, 4-38
 - delete release, 5-6
 - delete system, 4-38
 - delete task, 6-26
 - delete tlist, 6-26
 - delete version, 2-39
 - Edit File, 2-36
 - edit form, 6-25
 - edit monitor, 6-26

- edit release, 5-6
- edit task, 6-11, 6-25
- edit thread, 3-13, 3-28, 4-32
- examine build, 3-20 to 3-28, 5-5
- examine form, 6-25
- examine release, 5-6
- examine task, 6-15, 6-25
- examine thread, 3-13, 3-28, 4-6
- examine tlist, 6-9, 6-21, 6-25
- export, 3-23, 3-27, 4-15, 4-30
- fetch, 2-39
- help, 1-9 to 1-15
- merge, 2-25 to 2-32, 2-37
- name version, 2-32, 2-37, 4-18
- obsolete, 2-38
- online help about, 1-9 to 1-15
- promote, 4-37
- protect library, 2-38
- protect tlist, 6-26
- purge pool, 4-38
- pwd, 2-36
- quit, 1-18
- read, 2-13, 2-37
- recover library, 2-38
- recover monitor, 6-26
- recover pool, 4-38
- recover releases, 5-6
- recover tlist, 6-26
- recover user, 2-39
- reformat library, 2-39
- rename branch, 2-38
- rename element, 2-38
- replace, 2-16, 2-30, 2-35, 6-20
- reserve, 2-15 to 2-39, 6-20
- set builder, 4-12 to 4-13, 4-37
- set environment, 2-38
- set library, 2-6, 2-36
- set model, 3-8 to 3-28
- set system, 3-2 to 3-28
- set task, 6-26
- set thread, 3-11, 3-12, 3-28
- set tlist, 6-9, 6-25
- share, 2-39
- shell, 2-6, 2-36
- show branches, 2-39
- show builder, 4-38
- show builds, 3-20 to 3-28
- show derivation, 2-31, 2-35
- show elements, 2-10 to 2-11, 2-35
- show environment, 2-38
- show history, 2-14, 2-24, 2-35
- show model, 4-38
- show monitors held, 6-26
- show monitors owned, 6-26
- show pools, 4-38
- show releases, 5-6
- show reservations, 2-39
- show status, 2-38
- show system, 4-38
- show task, 6-26
- show tlist, 6-27
- show users, 2-39
- show version, 2-33
- tag task, 6-26
- watch tlists, 6-27
- wd, 2-3, 2-36

DSEE documentation, 1-17 to 1-18

dsee_examples directory

- contents, 1-16
- installing, 1-6

DSEE facility
 capabilities, 1-2
 context banner, 2-6, 3-3,
 6-4
 current settings, 1-8
 invoking, 1-7
 overview, 1-2
 purpose, 1-1
 sample session description,
 1-16 to 1-17
 terminating a session, 1-18
 user classes, 2-3
 version required by sample
 session, iv
 window
 context banner, 2-12
 context banners, 6-9

DSEE limits, 4-12

E

Edit File command, 2-36
 edit form command, 6-25
 edit monitor command, 6-26
 edit release command, 5-6
 edit task command, 6-11, 6-25
 edit thread command, 3-13,
 3-28, 4-32 to 4-34

Element blocks, 3-5, 3-7, 4-6

elements
 branches, creating, 2-22 to
 2-24
 building with reserved
 elements, 4-14
 cancelling reservation of,
 2-37
 comparing two versions of,
 2-19
 creating, 2-20 to 2-21
 creating new versions, 2-16

deactivating lines of descent,
 2-38
 debugging, 4-9 to 4-11
 definition, 1-2
 deleting, 2-34, 6-20
 deleting branches of, 2-37
 deleting versions of, 2-39
 displaying an icon for, 2-6 to
 2-9
 displaying branch leaf names,
 2-39
 displaying information about,
 2-10
 displaying reservations, 2-39
 displaying source reference
 environment for, 2-38
 displaying status of, 2-38
 examining, 2-13
 examining histories, 2-14 to
 2-15
 examining history of, 2-24
 examining history of
 (graphically), 2-31
 histories, 2-14 to 2-15
 lines of descent, 2-1
 listing, 2-10
 merging lines of descent,
 2-25 to 2-32
 modifying, 2-15 to 2-20
 monitoring, 6-16 to 6-24
 naming versions of, 2-32 to
 2-33
 protection of, 2-1
 reactivating a line of descent,
 2-37
 reading versions of, 2-13,
 2-39
 renaming, 2-38, 6-20
 renaming branches, 2-38
 replacing, 2-16
 replacing after merger, 2-30
 reserving, 2-15 to 2-20
 setting source reference
 environment, 2-38

- showing monitors on, 6-26
- summary of commands, 2-35
 - to 2-39
- using, 2-12 to 2-34
- versions. *See* versions,
 - version names
- watched by monitors, 6-20
- end** keyword, 3-4
- end of keywords**, 3-4
- environment** declaration, 3-8
- environment variables in system model, 3-8
- equivalences, 4-26 to 4-31
 - for many components, 4-27
 - function of, 4-26
 - lifespan of, 4-27
 - specifying, 4-26 to 4-31
 - types, 4-26
 - where stored, 4-29
- error messages, from compiler, 4-9
- exact** flag, 4-33
- examine build** command, 3-20
 - to 3-28, 5-5
- examine form** command, 6-25
- examine release** command, 5-6
- examine task** command, 6-15, 6-25
- examine thread** command, 3-13, 3-28, 4-6
- examine tlist** command, 6-9, 6-21, 6-25
- examples for sample DSEE session, 1-16
- export** command, 3-23, 3-27, 4-15, 4-30
- exports subdirectory, 5-3

- External blocks, 3-5, 3-7
- external** keyword, 3-6

F

- fetch** command, 2-39
- file as component dependency, 3-6
- flags
 - exact**, 4-33
 - when_exists**, 4-33
- forms
 - creating, 6-25
 - definition, 1-4
 - deleting, 6-25
 - editing, 6-25
 - examining, 6-25
- full build name, changed after promotion, 4-17
- full build names
 - abbreviated form for latest build, 4-15
 - definition of, 3-17
 - displaying, 3-20
 - displaying an icon for, 3-23, 4-18, 4-25
 - obtaining, 3-21, 3-23
 - using in equivalences, 4-27
 - using released builds, 5-5
 - using to name versions, 4-17

H

- header of a build map, 3-20
- help** command, 1-9 to 1-15
- Help menu, 1-9

history
 examining element history,
 2-14, 2-24
 library history, 2-12
Hold command, 1-15
host_type, identifying in system
 model, 3-8
host_type declaration, 3-8

I

icons
 copying an icon name, 4-26
 display area for, 1-8
 displaying
 branch icons, 2-16
 build icons, 3-25, 4-18,
 4-25
 build unit icons, 3-24
 element icons, 2-6 to
 2-9
 removing, 2-34
IDs for
 blocks, 3-4, 3-5, 3-7, 3-17
 libraries, 4-6
include file dependencies, 3-7
include files, changes to, 4-9
initial, version
 of a branch, 2-23
 of an element, 2-20
input, redirecting, 4-3
installing the sample system, 1-6
interrupting DSEE commands,
 1-15

K

keywords
 aggregate, 3-6
 alias, 3-8
 default, 3-7
 default for, 3-7
 depends_result, 3-7
 depends_source, 3-7
 depends_tools, 3-7
 element, 3-5
 end, 3-4
 end of, 3-4
 environment, 3-8
 external, 3-6
 host_type, 3-8
 library, 3-8
 model, 3-5
 %option, 4-32
 shell, 3-8
 system, 3-8
 title, 3-8
 translate, 3-6

L

language
 for configuration threads,
 3-10 to 3-14
 for equivalences, 4-27, 4-29
 for system models, 3-4 to
 3-9
libraries
 administrator of, 2-3
 and monitors, 6-16
 cleaning incomplete
 operations, 2-39

- contents, 2-10 to 2-11
- controlling concurrent access to, 2-39
- creating, 2-11 to 2-12
- current
 - definition, 2-3
 - setting, 2-3
- declaring in system model, 4-6
- definition, 1-3, 2-3
- deleting, 2-38
- examining the users of, 2-39
- history, 2-12
- identifying in system model, 3-8
- IDs in system model, 4-6
- listing the contents of, 2-10 to 2-11
- member of, 2-3
- non-user of, 2-3
- protecting, 2-38
- protection attributes, 2-1, 2-3
- reader of, 2-3
- recovering after crash, 2-38
- reformatting for new software, 2-39
- setting, 2-12
- showing monitors on elements in, 6-26
- specifier, 4-6
- summary of commands, 2-36 to 2-39
- using, 2-3 to 2-12

- library declaration, 3-8
- limit parameter, 3-20
- limits, 4-12
- lines of descent, 2-1
 - See also* branches
 - deactivating, 2-38
 - deleting, 2-37
 - displaying leaf names, 2-39

- merging, 2-25 to 2-32
- reactivating, 2-37
- renaming, 2-38
- target line of descent (in merger), 2-25

M

- main line of descent, 2-22, 2-25
- member of a library, 2-3
- menus
 - Help, 1-9
 - Misc., 1-14
 - transcript area, 1-15
- merge command, 2-25 to 2-32, 2-37
 - serial option, 2-27
 - query box, 2-28
 - replacing element after merge, 2-30
- merge_text, 2-26 to 2-32
 - description, 1-17
- merge_text/bugs branch, 2-26
- mine element, 2-21, 2-34
- Misc. menu, 1-14
- model. *See* system models
- Model blocks, 3-5
 - declarations, 3-8
 - definition of, 3-6
- model keyword, 3-6
- model threads, 1-18, 3-14
- monitor commands, command summary, 6-24 to 6-27
- monitors, 6-16 to 6-24
 - activating, 6-20 to 6-23
 - activation list, 6-17 to 6-27

- activation string, 6-18
- creating, 6-16 to 6-20
- definition, 1-4
- deleting, 6-24
- deleting after failure, 6-26
- displaying, 6-26
- editing task templates, 6-26
- libraries associated with, 6-16

my_library library, 2-11

N

name version command, 2-32

- to 2-33, 2-37, 4-18
- query box, 2-33

named versions, from builds, 4-17 to 4-21

names. *See* version names

names of

- branches, 2-22, 2-23
- builds, 3-17
- element versions, 2-32

See also version names

nodes, using several in builds, 4-11 to 4-14

noncritical option, 4-32

non-user of a library, 2-3

O

obsolete command, 2-38

%option keyword, 4-32

order program, introduction, 1-16

order.h, description, 1-16

order.sml, description, 1-17

order_main.c, description, 1-16

order_sub1.c, description, 1-16

order_sub2.c, description, 1-16

order_sub3.c, description, 1-16

order_sub4.c, description, 4-3

order_sys directory,

- introduction, 1-16

overrides, 4-26 to 4-31

- function of, 4-26
- types, 4-27

overview of DSEE, 1-2

P

parallel builds, 4-11 to 4-14

personal tasklist. *See* tasklists, personal

pools. *See* binary pools

position indicator, 1-12

prerequisites for a build, 3-2

primary source dependency, 3-6, 4-1

program

- components, 3-4
- modifying, 4-2 to 4-4
- rebuilt, 4-11 to 4-21
- releasing, 5-1

promote command, 4-37

promoting builds, 4-8, 4-16 to 4-17

- hierarchy of promotion, 4-17

promoting derived objects, and changed build names, 4-17

protect library command, 2-38

protect tlist command, 6-26

protection
 classes, 2-3
 of DSEE libraries, 2-1, 2-3
protection of elements, 2-1
purge pool command, 4-38
pwd command, 2-36

Q

qualified default declaration, 3-7
-query option (to build
 command), 4-22
quit command, 1-18

R

read command, 2-13, 2-37
reader of a library, 2-3
recover library command, 2-38
recover monitor command,
 6-26
recover pool command, 4-38
recover releases command, 5-6
recover tlist command, 6-26
recover user command, 2-39
reference node, 4-12
reformat library command,
 2-39
release directory. *See* releases
releases
 adding to, 5-6
 creating, 5-1 to 5-5
 deleting, 5-6

 displaying information about,
 5-6
 examining, 5-6
 introduction, 1-6
 managing, 5-1 to 5-6
 referring to in configuration
 threads, 5-5
 referring to in full build
 names, 5-5
 structure of, 5-3
 updating information on, 5-6
removing icons, 2-34
removing text, 4-28
rename branch command, 2-38
rename element command,
 2-38
replace command, 2-16, 2-30,
 2-35, 6-20
required
 settings, 1-8
 software for sample session,
 iv
reserve command, 2-15 to
 2-39, 6-20
 cancelling, 2-37
 displaying reserved elements,
 2-39
reserved elements, building with,
 4-14
reserved pool, 4-8
reserved pools, 4-10, 4-14,
 4-16
 promoting from, 4-8, 4-16
 to 4-17
 promoting objects from, 4-37
-reserved version rule, 3-11

reserved versions, building with,
4-4
result dependencies, 3-7

S

sample DSEE session,
introduction, 1-16
sample_library library, initial
contents, 1-16
Save command, 1-15
scripts, 4-3, 4-4, 4-9
scroll bars, 1-10
Search command, 1-15
-serial option (compare, merge
commands), 2-19, 2-27
server process manager, 4-13
server processes, needed for
running DSEE, iv
set builder command, 4-12 to
4-13, 4-37
set environment command,
2-38
set library command, 2-6, 2-36
set model command, 3-8 to
3-28
set system command, 3-2 to
3-28
set task command, 6-26
set thread command, 3-11,
3-12, 3-28
set tlist command, 6-9, 6-25
settings
current configuration thread,
3-12
current library, 2-3
current model thread, 3-14
current system, 3-2
current system model, 3-8 to
3-28
current task, 6-2, 6-3, 6-4
current tasklist, 6-3
share command, 2-39
shell command, 2-6, 2-36
shell commands
executed by monitors, 6-19
executing from within DSEE
environment, 2-6
shell declaration, 3-8
shells
creating with a source
reference environment,
4-37
identifying in system model,
3-8
show branches command, 2-39
show builder command, 4-38
show builds command, 3-20 to
3-28
show derivation command,
2-31, 2-35
show elements command, 2-10
to 2-11, 2-35
-full option, 2-10
show environment command,
2-38
show history command, 2-14 to
2-39
show model command, 4-38
show monitors held command,
6-26
show monitors owned
command, 6-26

- show pools** command, 4-38
- show releases** command, 5-6
- show reservations** command, 2-39
- show status** command, 2-38
- show system** command, 4-38
- show task** command, 6-26
- show tlist** command, 6-27
- show users** command, 2-39
- show version** command, 2-33
- source dependencies, 3-7, 4-1
- source dependency, 3-6
- source management, 2-1 to 2-39
- source reference environment, 4-37
 - displaying, 2-38
 - setting, 2-38
- source version, 2-25
- spm** (server process manager), 4-13
- substitute version specification, 4-27
- substitute build specification, 4-27
- system declaration, 3-8
- system directories, introduction, 1-5
- system directory. *See* systems
- system model, block, 3-4 to 3-9
 - block types, 3-5 to 3-6
- system model language, 3-4 to 3-9

- system models, 1-6, 3-3 to 3-9
 - block types
 - Aggregate, 3-6
 - Element, 3-7
 - External, 3-7
 - Model, 3-6
 - blocks
 - abbreviated form, 3-7
 - definition of, 3-4
 - ID, 3-6 to 3-7
 - keywords, 3-5 to 3-7
 - conditional processing of, 3-14
 - declarations, 3-6 to 3-8
 - aggregate**, 3-6
 - alias**, 3-8
 - default**, 3-7
 - depends_result**, 3-7
 - depends_source**, 3-7
 - depends_tools**, 3-7
 - element**, 3-6
 - element** (abbreviated form), 3-7
 - environment**, 3-8
 - external**, 3-6
 - external** (abbreviated form), 3-7
 - host_type**, 3-8
 - library**, 3-8
 - model**, 3-6
 - shell**, 3-8
 - system**, 3-8
 - title**, 3-8
 - translate**, 3-6
 - definition, 1-4
 - displaying information about, 4-38
 - how used during builds, 3-19
 - identifying title, 3-8
 - modifying for new components, 4-4

- parts of, 3-3
- setting current, 4-6
- setting current model, 3-8 to 3-9
- stored in multiple elements or files, 3-14
- translation rule, 3-6, 3-10, 3-20
- validating, 3-8, 4-6
- writing, 1-18
- systems, 3-2 to 3-3
 - building, 3-15 to 3-18
 - creating, 4-38
 - definition of, 3-2
 - deleting, 4-38
 - displaying information about releases of, 5-6
 - displaying the current system setting, 4-38
 - identifying in system model, 3-8
 - introduction, 1-5
 - setting current system, 3-2 to 3-3

T

- tag task command**, 6-26
- target line of descent, 2-25
- task editor, 6-4, 6-5 to 6-8
 - aborting, 6-8
 - exiting, 6-8
 - invoked by **create monitor**, 6-17
 - menus, 6-6 to 6-8
- task title, 6-2
- task transcript, 6-2
- tasklists, 6-2 to 6-16
 - active tasklist (library), 6-3
 - adding tasks to, 6-26
 - as organizational tools, 6-3
 - command summary, 6-25 to 6-27
 - creating, 6-3, 6-26
 - current tasklist, 6-3
 - definition, 1-3
 - deleting, 6-26
 - displaying name of current, 6-27
 - examining, 6-9 to 6-10
 - identifying for **alarm_server**, 6-27
 - library tasklist, 6-3
 - master tasklist, 6-3
 - numbering of tasks on, 6-10
 - personal tasklist, 6-3, 6-16, 6-21
 - and monitors, 6-19
 - examining, 6-9 to 6-10
 - protecting, 6-26
 - setting current, 6-9
 - updating task references on, 6-26
 - using, 6-2
- tasks, 6-2 to 6-16
 - active items
 - changing order of, 6-7
 - checking off, 6-11
 - creating, 6-6
 - definition, 6-2
 - adding to tasklists, 6-26
 - advantages of, 6-3
 - and history facility, 6-12
 - automatically created by monitors, 6-16
 - command summary, 6-25 to 6-26
 - creating, 6-3 to 6-5
 - creating forms for, 6-25
 - current task, 6-3
 - definition, 1-3
 - deleting, 6-26
 - deleting forms for, 6-25

- displaying information about, 6-26
- editing, 6-4, 6-5 to 6-8
- editing forms for, 6-25
- examining forms for, 6-25
- parts of, 6-2
- recording events in, 6-10 to 6-16
- referring to by number, 6-10
- setting current task, 6-26
- storage of, 6-11
- tagging element version with a task, 6-26
- task transcript, 6-10 to 6-16
 - automatic entries in, 6-12 to 6-16
- title, definition, 6-2
- transcript, definition, 6-2
- updating references to on tasklists, 6-26
- using, 6-2 to 6-16
- test**, description, 1-17
- test/bugfix branch**, 2-23
- text deletion, 4-28
- threads. *See* configuration threads, model threads
- title declaration**, 3-8
- tools dependencies, 3-7
- transcript area
 - clearing, 1-15
 - copying text, 1-12
 - holding, 1-15
 - menu, 1-15 to 1-16
 - searching for text, 1-15
 - writing to a file, 1-15
- translate keyword**, 3-6
- translation options, 3-10
 - specifying in configuration thread, 4-32

- translation rule, 3-6, 3-10
- translation rules, 3-20
 - obtaining information about execution, 4-38
 - options in, 4-32

U

- unqualified default declaration, 3-7
- update directory**, 4-3, 4-4, 4-9
 - information about, 1-16
- user classes, 2-3

V

- version names, 2-32 to 2-33
 - displaying all element versions with a name, 2-33
 - using in configuration threads, 4-33
- version of DSEE software, iv
- version specifications, named versions, 4-33
- versions
 - activating monitors when creating, 6-20
 - and tasks, 6-12 to 6-16
 - base version (in merger), 2-25
 - comparing differences between, 2-19
 - creating a shell using build versions, 4-37
 - deleting, 2-39
 - displaying source reference environment, 2-38

- identifying substitutes for builds, 4-27
- identifying those used in builds, 3-20
- most recent on main line of descent, 2-32
 - using in builds, 3-12
- naming, 2-32 to 2-33
- naming versions, 4-17 to 4-21
- numbers, 2-12
- reading, 2-39
- reserved, using in builds, 3-10, 3-11
- setting source reference environment, 2-38
- source version (in merger), 2-25
- specifying for builds, 3-10
- specifying named versions in threads, 3-10

- tagging with a task, 6-26
- using named versions in threads, 4-33

W

- watch tlists** command, 6-27
- wd** command, 2-3, 2-36
- when_exists** flag, 4-33
- wildcard specifications, 2-33, 3-7, 5-2, 6-17
- window (DSEE)
 - context banner, 6-4
 - context banners, 6-9
- working context, 3-2
- working directory, 2-36
 - setting, 2-3 to 2-39



Reader's Response

Please take a few minutes to give us the information we need to revise and improve our manuals from your point of view.

Document Title: *Getting Started with DSEE*

Order No.: 008788-A01

User Profile

Your Name _____ Title _____

Company _____

Address _____

Telephone number (____) _____ Date _____

When you use the HP/Apollo system, what job(s) do you perform?

- Programming Application End User
 Hardware Engineering System Administration
 Other (describe) _____

Characterize your level of experience in using the HP/Apollo system:

- Experienced user (2+ yrs.) New user (6 mos. or less)
 Moderately experienced user (6 mos.-2 yrs.)

What programming languages do you use with the HP/Apollo system?

Distribution

How do you know what manuals are available to support the products you're using or want to use?

What is a major concern for you in ordering books?

How would you evaluate this book?

	Excellent	Average	Poor
Completeness	1 2	3 4	5
Accuracy			
Usability			
Additional Comments:			

cut or fold along dotted line

old

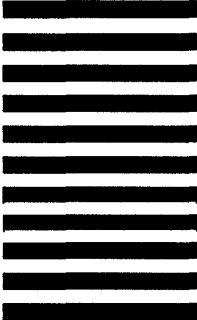


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

**Apollo Systems Division
Hewlett-Packard Company
Technical Publications
P.O. Box 451
Chelmsford, MA 01824**



old

Reader's Response

Please take a few minutes to give us the information we need to revise and improve our manuals from your point of view.

Document Title: *Getting Started with DSEE*

Order No.: 008788-A01

User Profile

Your Name _____ Title _____

Company _____

Address _____

Telephone number (____) _____ Date _____

When you use the HP/Apollo system, what job(s) do you perform?

- Programming Application End User
 Hardware Engineering System Administration
 Other (describe) _____

Characterize your level of experience in using the HP/Apollo system:

- Experienced user (2+ yrs.) New user (6 mos. or less)
 Moderately experienced user (6 mos.-2 yrs.)

What programming languages do you use with the HP/Apollo system?

Distribution

How do you know what manuals are available to support the products you're using or want to use?

What is a major concern for you in ordering books?

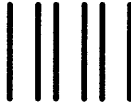
How would you evaluate this book?

	Excellent	Average	Poor
Completeness	1 2	3 4	5
Accuracy			
Usability			
Additional Comments:			

No postage necessary if mailed in the U.S.

cut or fold along dotted line

old

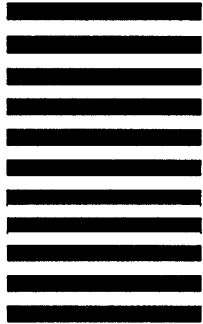


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

**Apollo Systems Division
Hewlett-Packard Company
Technical Publications
P.O. Box 451
Chelmsford, MA 01824**



old