
User's Guide

Copyright © 1992 by Rational

Product Number: 4000-00722

Rev. 1.0, November 1992 (Software Release 1_0_0)

This document is subject to change without notice.

Note the Reader's Comments forms at the end of this book, which request the user's evaluation to assist Rational in preparing future documentation.

AIX and RISC System/6000 are trademarks and IBM is a registered trademark of International Business Machines Corporation.

OSF/Motif is a trademark of Open Software Foundation, Inc.

Rational and R1000 are registered trademarks and Rational Environment and Rational Subsystems are trademarks of Rational.

UNIX is a registered trademark of UNIX System Laboratories.

X Window System is a trademark of MIT.

Rational, 3320 Scott Boulevard, Santa Clara, California 95054-3197

Contents

PREFACE

xiii

Organization of This Guide	xiii
What You Should Already Know	xiv
What You Should Read in This Guide	xiv
Conventions Used in This Guide	xv
Window Terms	xv
Text Conventions	xv
Mouse Terms	xvi
Compatibility With Layered Products	xvi
Related Documents	xvi

1 GETTING STARTED

1

What Is Access?	1
Logging In through Access	1
Opening an Access Window	2
Logging Into the Environment	2
The Main Access Window	3
Performing Operations in Access Windows	5
Access Commands	5
Executing Mouse Commands	6
Executing Menu Commands	7
Working in the Environment Area	7
Environment Commands	9
Executing a Command from a Command Window	9
Executing Item-Operation Commands	10
Logging Out from Access	11
Getting More Information	11

2 USING SPECIAL FEATURES

13

Window-Control Buttons	13
User-Defined Buttons	14
Image Palette	15
Function Key Palette	16
Debugger Palette	16
Just-Do-It Mode	18

3 GETTING HELP

19

Obtaining Information from the Access Help Window	19
Getting Introductory Information about Access	20
Getting Information about the Online Help System	20
Finding Out What a Menu Contains	20
Getting Help on the Window-Control Button Panel	20
Getting Help on the Mouse	20
Getting Help on Keys	21
Getting Help on Key Bindings	21
Getting Help on Function Keys	21
Finding What Command a Key Executes	21
Finding Out What Version of Access You Are Using	21
Getting a List of Help Topics	22
Getting a List of Environment Commands and Packages	22
Searching through the List	24
Displaying a Subset of Topics in a Filtered List	24
Getting Help on an Environment Command	24
Getting Help on Errors	25
Displaying Ada Specifications	25

4 MANAGING ENVIRONMENT WINDOWS

27

Moving between Environment Windows	27
Moving within an Environment Window	27
Traversing in a Window Using a Mark	28
Making a Mark	28
Traversing to a Mark	28
Moving the Cursor to the Beginning of the Window Frame	28
Moving the Cursor to the End of the Window Frame	28
Changing the Size of an Environment Window	28
Shrinking a Window	28
Expanding a Window	28
Expanding the Current Window over the Next Frame	29
Expanding the Current Window over the Previous Frame	29
Expanding the Current Window to Full Size	29
Making All Major Environment Windows the Same Size	29
Splitting an Environment Window	29
Removing an Environment Window	30
Removing a Window Temporarily	30
Removing an Image Permanently	30
Locking or Unlocking an Environment Window	30
Locking a Window	30
Unlocking a Window	31
Setting the Number of Window Frames	31
Getting a List of Environment Windows	31
Redisplaying a Window from the Image Palette	32
Searching for a Window	32
Updating the Image Palette	33
Setting Up a Standard Set of Windows	33

Changing the Size of the User Area and Full Image List	33
Closing the Image Palette	33
Finding Windows with Uncommitted Changes	33
Saving and Restoring a Set of Windows	33
Saving a Set of Windows	34
Restoring a Set of Windows	34

5 TRAVERSING THE ENVIRONMENT

35

Viewing Any Object	35
Viewing an Object in the Current Context	35
Making the Object Appear in the Next Window to Be Replaced	35
Making the Object Appear in the Same Window	35
Viewing the Enclosing Library	36
Making the Library Appear in the Next Window to Be Replaced	36
Making the Library Appear in the Same Window	36
Viewing Your Home Library	36

6 BROWSING ADA PROGRAMS

37

Moving between the Specification and Body of an Ada Unit	37
Viewing a Unit's Parent	37
Showing Occurrences of a Defined Ada Name	37
Showing References to a Defined Ada Name	37
Showing Unused Ada Constructs	38
Getting the Definition of an Identifier	38
Viewing the Specification of an Environment Package	38

7 WRITING ADA PROGRAMS

39

Creating an Ada Unit	39
Creating a New Ada Unit	39
Creating a Body from a Specification Automatically	40
Building a Unit from a Text File	40
Creating a Subunit	41
Promoting Ada Units	41
Promoting an Ada Unit to the Next Unit State	42
Promoting Ada Units to a Specific State	42
Demoting Ada Units	43
Demoting to the Previous Unit State	44
Demoting Units to a Specific State	44
Selecting Parent/Child Items in an Ada Unit	45
Modifying Units	45
Adding to a Unit	46
In the Source State	46
In the Installed or Coded State	46
Incrementally Changing an Existing Unit	47
In the Source State	47
In the Installed or Coded State	47
Changing Code into a Comment	47
Changing a Comment into Code	47

Deleting Part of an Existing Unit	48
In the Source State	48
In the Installed or Coded State	48
Changing the Name or Kind of an Ada Unit	48
In the Source State	48
In the Installed or Coded State	48
Adding a Subprogram to a Package	49
In the Source State	49
In the Installed or Coded State	49
Making a Package or Subprogram Body into a Subunit	50
Making a Subunit In-Line in the Parent Unit	50
Saving Incomplete Units	50
Executing a Library-Level Program	50
Creating a Loaded Main Program	51

8 DEBUGGING
53

Using the Debugger Palette	53
Displaying the Debugger Palette	53
Closing the Debugger Palette	54
Starting the Debugger	54
Redisplaying the Environment's Debugger Window	54
Displaying the Program Being Debugged	55
Stopping the Debugger	55
Finishing and Killing the Debugging Job	55
Finishing and Detaching from the Debugging Job	55
Stepping through the Program	55
Stepping by a Specific Number of Steps	55
Stepping by Every Statement	55
Stepping without Stopping in Called Subprograms	56
Stepping to the Enclosing Subprogram	56
Stopping a Task	56
Stopping the Current Task	56
Stopping All Tasks	56
Continuing a Task	56
Continuing the Current Task	56
Continuing All Tasks	56
Using Breakpoints	57
Setting Breakpoints	57
Showing Breakpoints	57
Removing Breakpoints	57
Removing a Specific Breakpoint	57
Removing All Breakpoints	57
Displaying the Value of a Program Variable	58
Modifying Variable Values	58
Examining the Call Stack	58
Displaying the Call Stack	58
Displaying Source for a Call-Stack Frame	59
Displaying Parameters for a Call-Stack Frame	59
Displaying the Parameters for the Current Selected Object	59
Displaying Parameters for a Specific Object	59
Traversing from the Call Stack	59

Setting Up Exception Handling	60
Catching Exceptions	60
Catching Unlisted Exceptions	60
Catching Any Exception	60
Showing Exceptions	61
Returning to the Point of Program Suspension	61
Showing Information	61
Showing All Debugger Activities	61
Showing Libraries	61
Showing Task information	61
Showing All Tasks	61
Showing Stopped Tasks	61
Showing Held Tasks	61

9 CREATING AND MODIFYING TEXT FILES

63

Creating a File	63
Viewing a File	63
Viewing a File in the Current Library	63
Viewing a File Located Anywhere	64
Opening an Existing File for Editing	64
Saving a File	64
Closing a File	64
Saving a File without Closing	64
Reverting to the Previous Version	65
Setting Tabs	65
Checking Tabs	65
Setting Typing Modes	65
Setting Overwrite Mode	65
Setting Insert Mode	66
Setting Wordwrap (Fill Mode)	66
Changing the Wordwrap Column	66

10 EDITING TEXT

67

Moving the Environment Cursor with the Keyboard	67
Selecting Text	68
Selecting Text in the Environment	68
Selecting a Word	68
Selecting the Preceding Word	68
Selecting the Next Word	68
Selecting a Sentence	68
Selecting a Paragraph	68
Selecting an Arbitrary Region of Text	68
Selecting All in a File	69
Selecting the Parent or Child	69
Deselecting Text	69
Selecting Text with Motif	69
Selecting a Region of Text	69
Deselecting a Region of Text	70

- Copying Selected Text 70
 - Copying an Environment Selection 70
 - Copying a Motif Selection 70
 - Copying a Line of Text 70
- Moving Selected Text 70
- Searching for and Replacing Text 71
 - Searching for a String 71
 - Searching and Replacing a String 72
 - Searching and Replacing All Occurrences of a String 72
 - Search Options 72
- Deleting Text 73
- Transposing Text 73
 - Transposing Characters 73
 - Transposing Words 73
 - Transposing Lines 74
- Changing the Case of Text 74
 - Making Text Uppercase 74
 - Making a Word Uppercase 74
 - Making a Selected Region of Text Uppercase 74
 - Making Text Lowercase 74
 - Making a Word Lowercase 74
 - Making a Selected Region of Text Uppercase 74
 - Making Text Capitalized 75
 - Making a Word Capitalized 75
 - Making a Selected Region of Text Capitalized 75
- Filling a Region of Text 75
- Justifying a Region of Text 75
- Getting Line Information 75
- Saving Changes 76
 - Saving Changes One Image at a Time 76
 - Saving Changes in All Images in a Single Operation 76
 - Saving the Image and Closing the File 76

11 MANAGING LIBRARIES

77

- Controlling the Library Display 77
 - Toggling Information on Library Objects 77
 - Showing More Objects in the Library 78
 - Showing Fewer Objects in the Library 78
- Creating Nonsubsystem Libraries 78
 - Creating a Directory 79
 - Creating a World 79
- Destroying Objects 80
- Copying Objects 81
- Moving or Renaming Objects 82
- Printing Objects and Images 83
 - Printing Multiple Objects 84
 - Selecting the Printer 84
 - Specifying the Pages to Print 84
 - Printing All Pages in a File 84
 - Printing Specific Pages 84
 - Format Options 85

Page-Layout Options	85
Other Options	85

12 USING CMVC

87

Creating a Subsystem	87
Making a Path	89
Making a Subpath	90
Making a Spec View	91
Releasing Configurations	92
Making a Release View	92
Making a Configuration Release	94
Making a Code View	96
Creating a System	96
Making Objects Controlled or Uncontrolled	97
Checking Out an Object for Changes	98
Checking In an Object After Changes	98
Accepting Changes	99
Accepting Changes from a View	99
Accepting Changes If the Destination Is an Object	99
Accepting Changes If the Destination Is a View	101
Accepting Changes from an Object	101
Joining Objects in Different Views	101
Severing Objects in Different Views	102
Reverting to a Previous Generation	103
Creating a New Activity	104
Adding an Activity Entry	104
Starting the CMVC Editor	105
Collecting Information about Controlled Objects	105
Creating a Work Order	105
Creating a Work-Order List	106
Creating a Venture	107
Getting Information about a View	107
Getting the History of an Object	108

13 CONTROLLING JOBS

111

Displaying Current Jobs	111
Displaying Your Current Jobs	111
Displaying All Current Jobs	112
Disconnecting from a Job (Putting It in the Background)	112
Reconnecting to a Job (Putting It in the Foreground)	113
Disabling a Job	113
Enabling a Job	114
Killing a Job	114
Killing the Current Job or the Last Job Created	114
Killing Any Job	114

14 CUSTOMIZING YOUR ACCESS WORKSPACE **117**

Executing Menu Commands with User-Defined Buttons	117
Creating a Button for a Menu Command	118
Changing the Size of the Button Area	118
Activating a User-Defined Button	118
Deleting Buttons	118
Saving Buttons	118
Building and Executing Macros	119
Defining a Macro	119
Executing a Macro	119
Binding a Macro to a Key	119
Saving the Current Macros	119
Rebinding Keys	120
Rebinding Temporarily	120
Rebinding Permanently	120
Changing the Screen to Inverse Video	120
Setting the Visual Bell	120

15 USING COMMAND WINDOWS **121**

Creating and Executing a Command-Window Program	121
Getting Command Completion	121
Moving in a Command Window	122
Moving to an Underline	122
Moving to a Prompt or Underline	122
Turning Off a Prompt	122
Turning Off Underlines	122
Reexecuting a Command	123
Entering a New Command	123
Entering a New Command in the Same Window	123
Clearing a Command Window of Unneeded Text	123
Going Back to Previous Commands	123
Redisplaying a Previous Command in the Historical Sequence	123
Redisplaying a Later Command in the Historical Sequence	124
Getting the Parameters of a Command Bound to a Key	124

A SETTING UP ACCESS **125**

How Access Works	125
X Application Components	125
Access as an X Application	126
Requirements for Running Access	127
Configurations for Using Access	127

B USER-INTERFACE BASICS **129**

Choosing Menu Commands	129
Terms for Describing Menus	129

Using the Mouse to Choose Commands from Menus	129
Clicking with the Mouse	129
Dragging with the Mouse	130
Using the Keyboard to Choose Commands from Menus	130
Using Mnemonics	130
Choosing Directly from the Menu Bar	131
Executing Window-Control Button Commands	131
Responding to Dialog Boxes	131
Terms for Describing Dialog Boxes	132
Using the Mouse to Respond to Dialog Boxes	133
Text-Entry Boxes	133
Other Controls	134
Using the Keyboard to Respond to Dialog Boxes	134
Navigating a Dialog Box	134
Specifying Information	135
Initiating the Action of a Command Button	135
Shortcut for Canceling a Dialog Box	136

C ACCESS EQUIVALENTS: ENVIRONMENT COMMANDS	137
---	------------

D ACCESS EQUIVALENTS: KEY BINDINGS	151
---	------------

Fundamental Set of Logical Keys	152
Object Operations	155
Region Operations	157
Window Operations	160
Image Operations	163
Line Operations	165
Word Operations	167
Mark Operations	169

INDEX	173
--------------	------------

Preface

This preface describes the organization of the *Rational Access User's Guide*, suggests appropriate sections for various users to read, and outlines the text and mouse conventions used in this guide.

ORGANIZATION OF THIS GUIDE

The *Rational Access User's Guide* is divided into these tabbed sections:

- **Getting Started:** Chapters 1 through 3 introduce Access and graphical user interfaces and discuss system information and terminology used in this guide.
- **Performing Environment Operations:** Chapters 4 through 15 describe, with simple step-by-step procedures, how to perform common operations in the Rational Environment™ using the Rational Access graphical user interface.
- **Appendixes:**
 - Appendix A, “Setting Up Access,” describes sample machine configurations for running Access.
 - Appendix B, “User-Interface Basics,” describes Motif™-style user-interface basics.
 - Appendix C, “Access Equivalents: Environment Commands,” lists basic Environment commands and the Access menu items and buttons that are most similar to them.
 - Appendix D, “Access Equivalents: Key Bindings,” lists the logical key names used in Rational’s training and documentation and the Access key bindings, mouse buttons, menu items, and menu buttons that are most closely related to them.
- **Quick Reference:** Quick reference for Access key and mouse bindings.
- **Index**

This guide focuses on using Access to perform basic operations on Ada programs and text files in single libraries. Some of the areas are: executing commands, managing windows, writing and debugging programs, and editing text files. Areas not included are multibrary development, sophisticated use of Rational Subsystems™, and optional products such as the Rational Design Facility, Rational Mail Utility, and host-target development products.

Note that this guide does not provide tasks for all Access commands.

WHAT YOU SHOULD ALREADY KNOW

To follow the instructions in this guide, you need to:

- Know basic Environment concepts, including:
 - Environment library hierarchy
 - The Environment's compilation model, including unit states
 - Subsystem (CMVC) theory
 - Basic use of the debugger
- Know how to perform basic UNIX[®] operations in the UNIX shell you are using (required for Access setup and startup), including how to:
 - Log into a UNIX workstation
 - Execute UNIX commands
 - Edit a file
- Have a general understanding of:
 - Your X Window System™ (X) environment
 - Basic operations common to mouse and menu-driven user interfaces
 - Basic window operations provided by the window manager that you are using (for example, the Motif Window Manager, **mwm**)

See "Related Documents," below, for documents that can provide you with this prerequisite information.

WHAT YOU SHOULD READ IN THIS GUIDE

The *Rational Access User's Guide* is written to assist new Environment users, current Environment users who are new to Rational Access, and new and experienced system administrators.

This guide assumes some familiarity with the Rational Environment.

- If you are familiar with Motif-style interfaces and the Environment but new to Access, read Chapters 1 and 2, and refer to Chapters 3–15 for step-by-step task information. Also see Appendix C, "Access Equivalents: Environment Commands," for information about direct correlations between Environment commands and Access menu items and buttons, and Appendix D, "Access Equivalents: Key Bindings," to find out where in the Access interface you can find familiar logical keys, such as the item-operations.
- If you are new to Motif-style interfaces, read chapters 1 and 2 and Appendix B, "User-Interface Basics." Refer to Chapters 3–15 for step-by-step task information.
- If you are familiar with Motif but not the Environment and Access, read Chapter 2, "Using Special Features," and refer to Chapters 3–15 for step-by-step task information. While learning the Environment, you may also want to refer to Appendix D, "Access Equivalents: Key Bindings," to find out where in the Access interface you can find the logical keys referred to by Rational's documentation and training.
- If you are a system administrator responsible for installing Access and supporting Access users, read Chapter 1, "Getting Started," and Appendix A, "Setting Up

Access.” Also consult the Rational Access Installation Note and Rational Access Release Information.

CONVENTIONS USED IN THIS GUIDE

The following subsections describe window, text, and mouse terminology.

Window Terms

This guide uses examples of Access windows and dialog boxes running the MIT X11 Server. For information on manipulating the X window frames, see the *X Window System User's Guide* (see “Related Documents,” below).

Note that *Access window* refers to the entire Access-window area, including the menu bar, button panels, and Environment area. *Environment window* refers to one of the individual windows (there are three by default) inside the Access window in which text files, Ada units, and output appear and where you perform Environment operations. Window terms are discussed in more detail in Chapter 1, “Getting Started,” and Appendix B, “User-Interface Basics.”

Text Conventions

The following table lists and defines examples of text conventions used in this guide.

Example	Meaning
<i>slider</i>	Indicates new terms where they are defined
<i>R1000-name</i>	Indicates text you must type in a UNIX command line or in a command window; for example, enter the name of your R1000
mwm	Indicates a UNIX command
Rational	Indicates literal characters that you type or see in a dialog box, a file, an Ada unit, a UNIX command line, or an Environment command or message window.
Text.Create	Indicates Ada identifiers (including Environment command names and pathnames) in body text
Edit:Cut	Identifies a command on an Access menu; for example, Edit:Cut refers to the Cut command on the Edit menu
OK	Identifies command buttons on dialog boxes
[Return]	Represents a key that must be pressed to initiate or complete an action
[Control] [G]	Represents keys that must be pressed simultaneously; for example, while holding down [Control], press [G]
<i>Note:</i>	Indicates important, additional information; text is in italics
<i>Alternative:</i>	Indicates keyboard alternatives to mouse or command window operations; text is in italics

Mouse Terms

The following table lists and defines terms used in mouse operations.

Term	Meaning
Pointer	Refers to the visible representation of the mouse on the screen; moving the mouse moves the pointer
Click	Press and release the left button without moving the pointer
Double-click	Click the left button twice in rapid succession
[Shift]+click	Hold down the [Shift] key while clicking the left button
[Control]+click	Hold down the [Control] key while clicking the left button
Drag	Hold down the left button while moving the pointer
[Shift]+drag	Hold down the [Shift] key and the left button while moving the pointer
[Control]+drag	Hold down the [Control] key and the left button while moving the pointer

Note: For operations using the right or middle mouse button, the text will specify which button to use.

COMPATIBILITY WITH LAYERED PRODUCTS

Supported layered products can be started, used, and exited using Access menu commands. Commands for supported layered products appear in the Tools menu.

If you try to operate a layered product that is not installed or authorized, a message appears in the message window saying that the product is not authorized.

For more information, see the Rational Access Release Information and consult online help for specific commands used in layered products.

RELATED DOCUMENTS

Because the *Rational Access User's Guide* discusses only basic Environment operations, you may want to refer to other materials for further information.

For information about Rational Access, see:

- Rational Access Installation Note
- Rational Access Release Information

For information about the Rational Environment, see:

- *Rational Environment User's Guide*
- *Rational Environment Reference Manual*
- *System Manager's Guide*

For information about the X Window System and OSF/Motif-based user interfaces, see:

- V. Quercia and T. O'Reilly, *X Window System User's Guide, OSF/Motif Edition* (Sebastopol, CA: O'Reilly & Associates, 1991).
- *OSF/Motif User's Guide* (Englewood Cliffs, NJ: Prentice-Hall, 1991).

1

Getting Started

This chapter describes Rational Access and a standard machine configuration for running Access. It also describes how to log into the Rational Environment™ and execute commands through Access.

WHAT IS ACCESS?

Rational Access is an OSF/Motif™-style graphical user interface to the Rational Environment, which is an environment for developing and maintaining large software projects written in Ada. Access allows you to perform standard Environment operations using such OSF/Motif and graphical user-interface tools as:

- A mouse
- Pull-down menus
- Dialog boxes
- Persistent control palettes
- Control buttons
- Extensive online help

Access also supports existing Environment paradigms, such as:

- Item operations
- Command windows

Access allows you to start, use, and exit supported layered products.

Access runs as an X Window System™ (X) application on a workstation. Access brings up the Environment's interface as a special-purpose X window (the *Access window*) on your display (a workstation monitor or an X terminal). Your system manager has determined which workstation and display you use to run Access. If you want to know more about how Access works or how it has been set up for you, see Appendix A.

LOGGING IN THROUGH ACCESS

You use UNIX® commands to start Access and connect to an R1000®. Below are basic steps for starting Access, including other useful UNIX options. Once you are logged into the Environment, all operations are performed using Access or Environment commands.

Note that you can customize characteristics of the Access window in your .Xdefaults file, including fonts, keyboard-focus policy, location, size, and colors.

Opening an Access Window

1. Begin in an X window that contains a UNIX command prompt.
2. Enter `rational`, including any UNIX command options. Three options are particularly useful with the **rational** command:
 - `-e`, which executes a command. To open an Access window and connect to an R1000 at the same time, follow the `-e` with the command `telnet R1000-name`.
 - `-geometry`, which allows you to set the size and position of the Access window without using the mouse. For example, entering the parameters `80x24+0+0` creates an Access window that is 80 columns wide and 24 lines long at coordinates (0,0)—that is, at the upper-left corner of the screen.
 - `-title`, which allows you to give a specific title to the Access window. This title will also identify the window if you turn it into an icon using the window manager, unless you separately specify an `-icontitle` option.

As an example, you could enter the following command:

```
> rational -title Debug -geometry 80x24+0+0 -e telnet R1000-name &
```

3. Press [Return].

Wait a few seconds; a Rational Access window will appear. This creates a standard-sized Access window (unless you specify a `-geometry` option) connected to the R1000.

Note: For more information about the **rational** command, see the **rational UNIX man page** (enter `man rational` at a UNIX prompt) or type `rational -help` when logging in at the UNIX command prompt.

Logging Into the Environment

The following steps are the same for all Rational interfaces.

1. Begin in an Access window that is connected to an R1000.

An Access window connected to an R1000 displays the following:

```
Trying 89.64.3.3 ...
Connected to R1000 name.
Escape character is '^]'.
```

If your Access window contains a UNIX prompt, enter `telnet R1000 name` and press [Return]. You will then see the display shown above.
2. Press [Return] until you get an `Enter user name:` prompt.
3. Enter your username and press [Return].
4. At the `Enter password:` prompt, enter your password and press [Return].
5. At the `Enter session name:` prompt, enter the name of the Environment session.
 - To log into your default session (`S_1`), press [Return].
 - To log into another session, enter the name of that session and press [Return].

If the session does not exist, the Environment will ask if you want it created. If you do, enter `y`; if not, enter `n`.

Note: If you take too long to input any of the entries, login will halt. Press [Return] to get a new Enter user name: prompt. Some systems may require you to reestablish the Telnet connection.

THE MAIN ACCESS WINDOW

Once you have logged in, you will see the main Access window. The main Access window, the starting point for your work in Access and the Environment, will look something like the one in Figure 1-1.

Note that your Access window may look different if your system administrator has customized your display. The examples in this book are based on the Motif Window Manager.

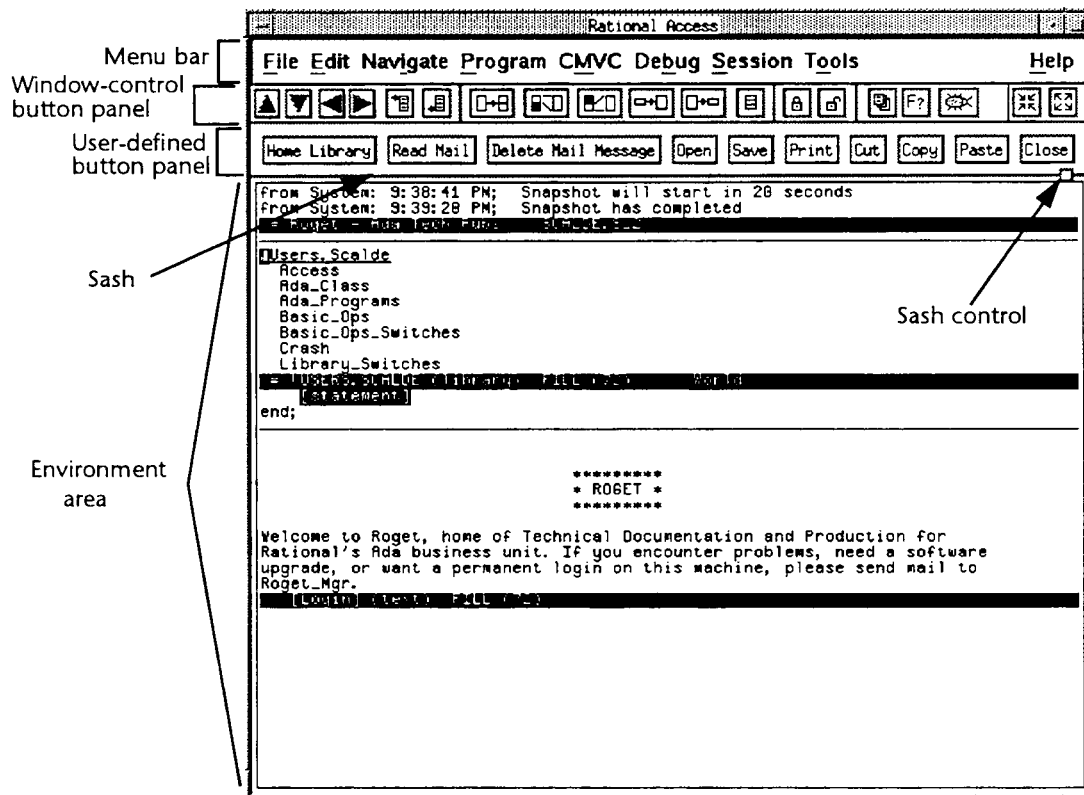


Figure 1-1 The Access Window

The Access window contains the following major elements:

Menu bar: Contains the nine Access pull-down menus. Underlined letters indicate the keyboard character that can be used in combination with the [Meta] key to activate the menu. Following is a list of the nine pull-down menus:

- The File menu is used for creating, accessing, copying, moving, and deleting Environment objects.
- The Edit menu is used for performing common editing operations, searching, and checking spelling.
- The Navigate menu is used for moving the Environment cursor between and within images.

- The Program menu is used for creating, changing, and compiling Ada programs.
- The CMVC menu is used for controlling, manipulating, and querying for information about objects in subsystems.
- The Debug menu is used for debugging Ada programs.
- The Session menu is used for customizing your Environment session or Access display. It provides access to Environment profiles, switches, and searchlists.
- The Tools menu is used for accessing Rational's layered software products and for Environment facilities that provide mail, operator capabilities, and system information.
- The Help menu is used for getting information about Rational Access and the Environment.

Window-control button panel: Contains the buttons, each labeled with a graphic, that control specific Environment window functions. Buttons are activated by placing the pointer on the button and clicking the mouse. Table 1-1 lists the buttons and their functions.

User-defined button panel: Contains user-defined buttons. You can create a button for any menu command. See Chapter 14, "Customizing Your Access Workspace," for information on how to create and execute buttons.

Sash: Separates the Environment area from the button panel.

Sash control: Controls the height of the sash.

Environment area: Defines where you do work. See "Working in the Environment Area," below, for information about the Environment area.

Table 1-1 Window-Control Buttons




















Button	Name	Function
	Scroll Up	Scrolls image up
	Scroll Down	Scrolls image down
	Scroll Left	Scrolls image left
	Scroll Right	Scrolls image right
	Top of Image	Moves cursor to beginning of image
	Bottom of Image	Moves cursor to end of image
	Copy Window	Splits window into two frames, each containing the same image
	Join Next Window	Joins current window with next
	Join Previous Window	Joins current window with previous
	Expand Window	Expands window four lines
	Shrink Window	Shrinks window four lines
	Realign Windows	Makes all windows the same size

Table 1-1 Window-Control Buttons (continued)

Button	Name	Function
	Lock Window	Makes the window unable to be replaced until user releases it
	Unlock Window	Makes the window able to be replaced
	Show Image Palette	Brings up the Image Palette
	Show Function Key Palette	Brings up the Function Key Palette
	Show Debugger Palette	Brings up the Debugger Palette
	Remove Window	Removes window from Environment area (this button also appears on the Access control palettes)
	Fully Expand Window	Expands window to full Environment area

PERFORMING OPERATIONS IN ACCESS WINDOWS

After you open an Access window, you perform Access and Environment operations by choosing commands from menus in the main window. Access is an OSF/Motif-based application, which means it follows OSF/Motif standards for menu and dialog-box operations:

- You use the mouse or mnemonic keys to choose commands from the menus.
- When the chosen menu commands bring up dialog boxes, you use the mouse and/or keyboard shortcuts to specify the requested information.

If you are relatively inexperienced with this kind of user interface, see Appendix B. For complete information, see the *OSF/Motif User's Guide*.

Operations that affect whole windows (such as moving and resizing) are controlled by the window manager you are using; see your window-manager documentation for details.

Note that you can control several characteristics of Access using X Window System resources. For details, see the UNIX **man** page for the **rational** command.

Access Commands

Access provides its own set of commands, called *Access commands*. These commands are what you see as menu items and buttons. Access commands perform operations specific to Access, such as saving user-defined buttons, as well as basic Environment operations, such as creating new files. Access commands are defined in package !Commands.Menu_Operations and are built on top of existing *Environment commands* (which are available on all Rational user interfaces). For example, the Text File command (found on the New submenu of the File menu), is the same as the Environment command Text.Create. **File:Open**, however, uses the Environment

command `Common.Edit` and, if necessary, `Cmvc.Check_Out`. For more information on Environment commands, see "Environment Commands," later in this chapter.

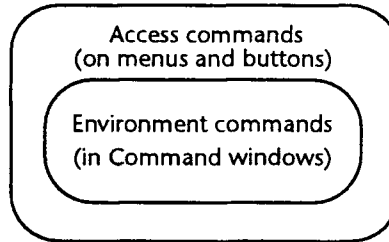


Figure 1-2 Relationship of Access Commands to Environment Commands

Key and mouse operations can be bound to either Access or Environment commands. For example, the [F8] key executes the Environment command `Common.Promote`, and the [F10] key changes the keyboard focus to the menu bar, which is an Access-specific operation.

Executing Mouse Commands

This book assumes that you are using a three-button mouse with the right-handed configuration, as shown below. If you have modified the button configuration on your mouse, you need to keep in mind what button you would actually use. You can use the mouse alone to perform some Access, Environment, and Motif operations, as shown in Table 1-2. Note that there are two kinds of text selection: Environment and Motif.

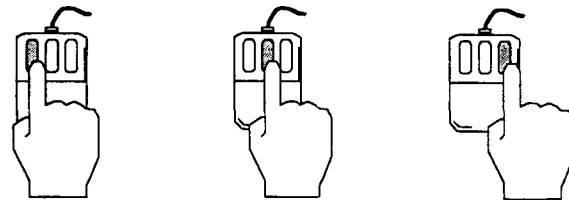


Table 1-2 Access Mouse Functions

Action	Left Button	Middle Button	Right Button
Click	Position Cursor	Motif Selection Copy	
[Shift]+click	Motif Selection End		
Double-click	Definition		Enclosing
[Shift]+double-click	Definition in Place		Enclosing in Place
[Control]+click	Region Start	Region Copy	Region End
[Control]+double-click	Select Object		Select Child
Drag	Motif Selection		
[Control]+drag	Region Selection		

Executing Menu Commands

This manual uses the notation **Menu:Command** to refer to commands on menus. For example, **Edit:Copy** refers to the Copy command on the Edit menu.

Some commands are executed from submenus. Notation for these commands is **Menu:Submenu:Command**. For example, **File:New:Text File** refers to the Text File command on the New submenu of the File menu.

To choose a command from a menu:

1. Put the pointer on the appropriate menu title in the menu bar and click. This opens the menu and displays it below the menu bar.
2. Put the pointer on the name of the desired item and click. This initiates the command. Note that:
 - An ellipsis (...) after a command name indicates that a dialog box will appear requesting further information.
 - An arrow after a command name indicates that a submenu will appear with more command choices.
 - A gray command name indicates that the command is currently inapplicable and cannot be chosen.

See "Choosing Menu Commands" in Appendix B for more information, including alternative mouse techniques and using the keyboard to choose a menu command.

WORKING IN THE ENVIRONMENT AREA

The Environment area of your Access window is where you perform all Environment operations, including creating and editing text files and Ada programs, debugging Ada programs, and creating and manipulating subsystems. The Environment area, as shown in Figure 1-3, is the same as in other Rational Environment interfaces (such as RXI and RWI). Thus, if you are familiar with another Environment interface, you may not need to read this section.

The Environment area contains the following elements:

Message window: Displays system and error information from the Environment. Some Environment operations, such as the [Prompt For] command, require input in the message window.

Message-window banner: Displays the name of your R1000, your username, and the name of the session you are logged into. In addition, whenever you execute a command or run a job in the foreground, the message-window banner indicates this by displaying the ...running message.

Environment windows: Display some part of an image of an Environment object, such as a library, text file, or an Ada unit, in which you can write and edit. You can scroll the image in all directions.

Environment-window banner: Provides information about the object displayed in the window above, such as its name and class. See Table 1-3.

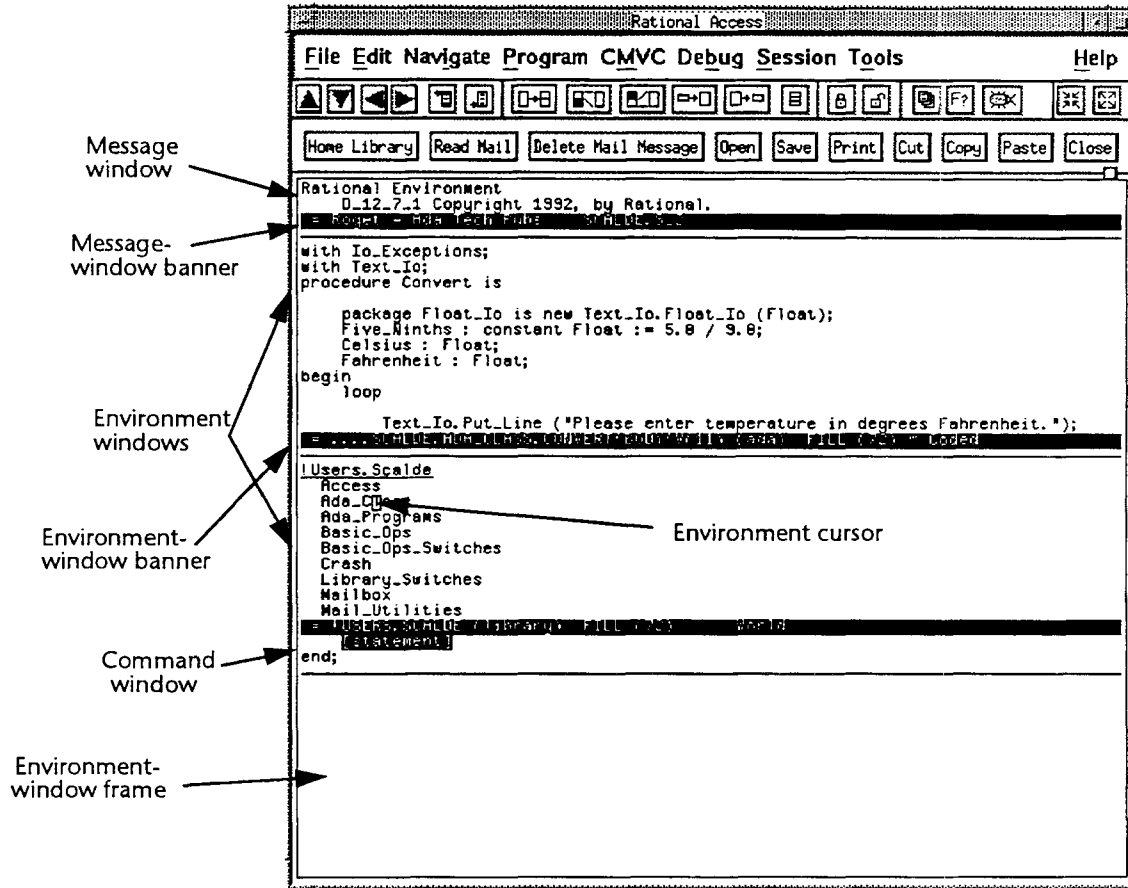


Figure 1-3 The Environment Area

Table 1-3 Symbols in Environment-Window Banner

Symbol	Definition
=	Indicates that the image is read-only
(blank)	Indicates that the image is modifiable and that no changes have been made since it was last saved
*	Indicates that the image is modifiable and that changes have been made to it since it was last saved
#	Indicates that the image of an Ada unit is modifiable and that the image has been changed since it was last saved
!	Indicates that the image is currently read-only because a job has obtained access to the object in that window

Environment cursor: Appears as a reverse-video rectangle around a character or space. It marks the text-insertion point within a window.

Command window: Allows you to enter and execute any Environment command. Command windows are attached to the major window above them.

Environment-window frame: Defines the screen space occupied by an Environment window, its banner, and any attached command windows. The available

screen is divided into several frames, within which windows can be placed. You can resize the window frames using the window control buttons. See Chapter 4, “Managing Environment Windows,” for specific information about manipulating Environment-window frames.

Mouse pointer (not shown): Marks the location of the mouse. It appears as an arrow or I-beam, depending on the location within the Access window, and as a watch when a job is executing.

Environment Commands

In addition to performing operations through the Access menus, you can use the Environment’s *command-window* interface. Command windows are special-purpose windows through which you execute *Environment commands*.

Environment commands are predefined Ada procedures and functions that are provided for your use. These subprograms are defined in packages, whose names reflect the various Environment objects and other functional groupings of Environment operations. The specifications of these packages are located in the !Commands and !Tools libraries.

Environment commands perform most of the operations available through Access, as well as other, more complex operations. Using Environment commands, you can create, display, and modify objects, manage windows, display information, and the like. Environment commands and command windows are especially useful for:

- Performing several operations in a single instruction
- Performing complex operations, such as copying multiple objects between machines

The use of Ada as the Environment’s command language provides a number of possibilities for entering commands in command windows. Because command windows contain Ada block statements, and because Environment commands are entered as calls to Ada procedures:

- The rules of Ada syntax allow alternative ways to specify parameter values for those commands that have parameters.
- You can declare variables, constants, and the like within command windows, and you can build multiple-line programs.

Command windows appear in the Environment area of the Access window and are available in all Rational user interfaces. Environment commands are documented in the *Rational Environment Reference Manual*. Access menu and button equivalents for many Environment commands are listed in Appendix C, “Access Equivalents: Environment Commands.”

Executing a Command from a Command Window

To execute a command from a command window:

1. Press [Cmd Window] to open a command window.
2. Enter the command or a unique fragment of the command name.
3. Press [Complete] to display any parameter prompts.
4. Enter any necessary parameter values.

5. Press [Promote] to execute the command.

See Chapter 15, "Using Command Windows," for more information.

Executing Item-Operation Commands

Item operation refers to particular key combinations that consist of a special kind of key, called an *item key*, followed by another key indicating the operation to be performed on that item.

This command paradigm is available on all Rational interfaces, including Access, although many Access commands offer alternatives to the item-operation technique. New Environment users should note that although Access provides alternatives to the item-operation paradigm, all the material in the *Rational Environment Reference Manual* and the *Rational Environment User's Guide* contains examples using item operations. See Appendix D, "Access Equivalents: Key Bindings," for a list of item-operation commands and corresponding Access commands.

Table 1-4 lists the seven item keys and their respective positions on an unmodified keyboard.

Table 1-4 Item Keys

Item Key	Location
Object	[Control][F1]
Region	[Control][F2]
Window	[Control][F3]
Image	[Control][F4]
Line	[Control][F5]
Word	[Control][F6]
Mark	[Control][F7]

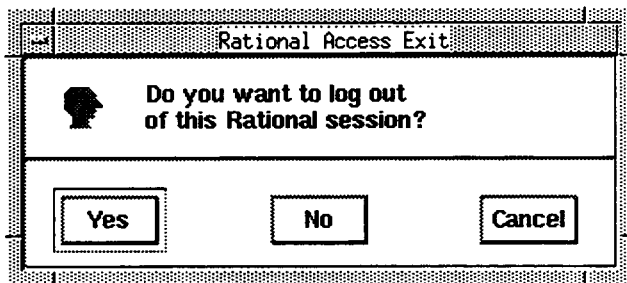
Operation keys can be main keyboard keys, function keys, auxiliary keys, or cursor keys. In general, commands that execute similar operations are bound to combinations containing a common operation key. For example, the combinations [Line] - [D], [Word] - [D], and [Window] - [D] all delete an item, as indicated by the shared operation key, [D].

For a complete list of operation keys, see Appendix D in this guide or open the Function Key Palette and click on one of the item keys. An active template will appear containing all the operation keys for that item key.

LOGGING OUT FROM ACCESS

1. Choose File:Exit.

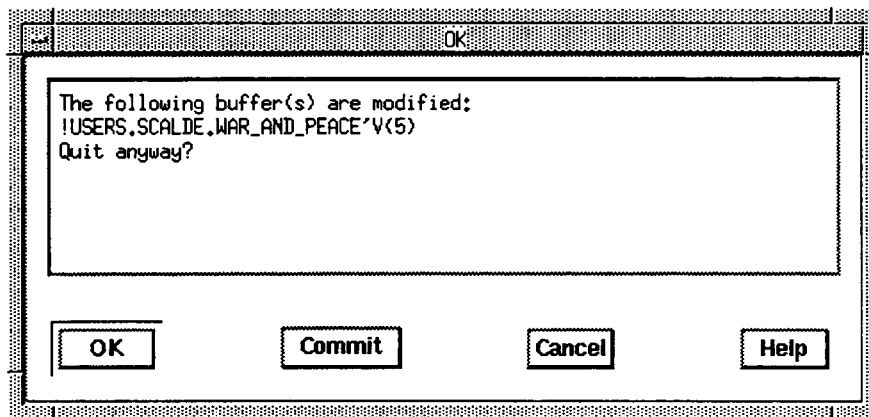
The following dialog box appears:



2. Click the Yes or No button.

- Click No to exit from Access but remain logged into your session. This is useful if the R1000 becomes nonresponsive; otherwise, you cannot exit Access through the Access window.
- Click Yes to log out.

If there are any uncommitted changes, a second dialog box appears:



Note that the dialog box lists the last three modified buffers brought up in Environment windows.

3. Click the OK or Commit button.
 - Click OK to log out without committing the changes.
 - Click Commit to save the changes before logging out.

The Access window disappears.

GETTING MORE INFORMATION

Rational Access offers extensive online help for Access and Environment commands (see Chapter 3, “Getting Help”). For more information on Environment commands and procedures, also see the *Rational Environment Reference Manual*. The rest of this guide offers more information about Rational Access:

- For information about specific, basic Environment operations using Access, see Chapters 3–15.
- For information about setting up configurations to run Access, see Appendix A.
- For information about basic graphical user-interface operations, see Appendix B.
- For a list of basic Environment commands and the Access menu items and buttons that are most closely related to them, see Appendix C, “Access Equivalents: Environment Commands.”
- For a list of the logical key names used in Rational's training and documentation, including the item-operation key combinations, and the Access key bindings, mouse buttons, menu items, and menu buttons that are most closely related to them, see Appendix D, “Access Equivalents: Key Bindings.”

2

Using Special Features

This chapter describes features specific to Access:

- Window-control buttons
- User-defined buttons
- Image Palette
- Function Key Palette
- Debugger Palette
- Just-Do-It mode

WINDOW-CONTROL BUTTONS

A panel of window-control buttons, shown in Figure 2-1, allow you to manage the visual characteristics of the Environment area in your Access window. See Chapter 1, "Getting Started," for a list of button names and their functions. To activate a button, place the pointer on the button and click.

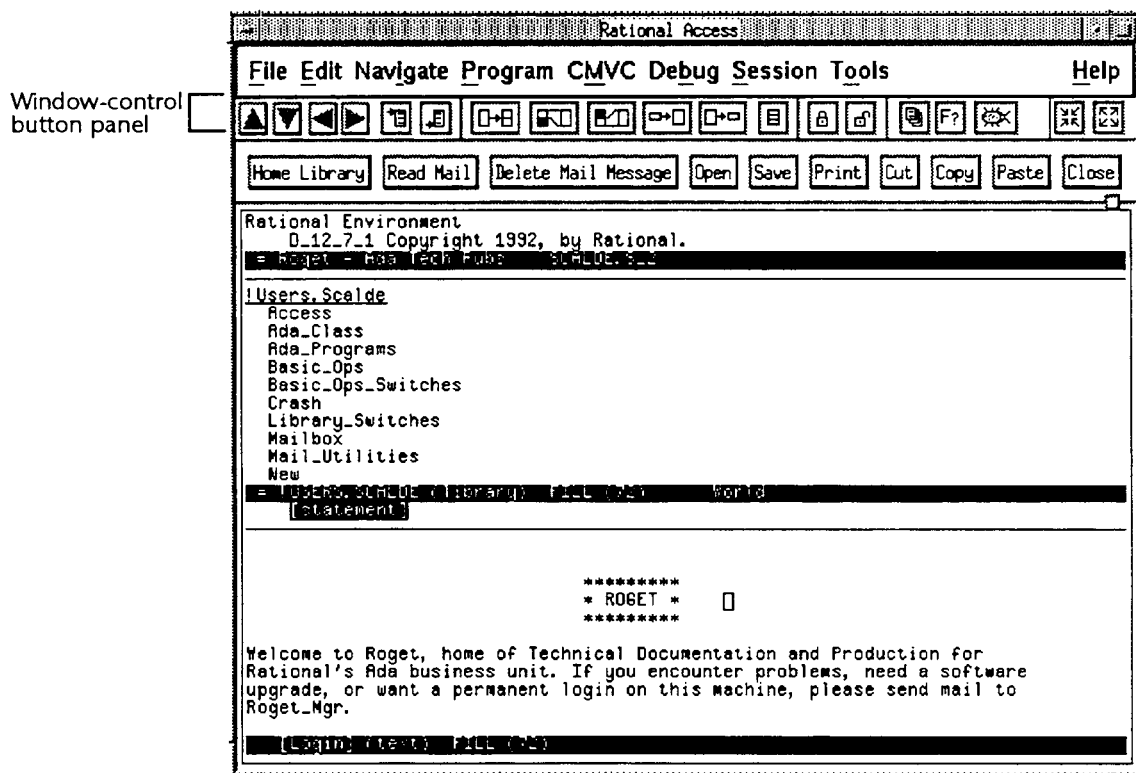


Figure 2-1 The Window-Control Button Panel in the Main Access Window

USER-DEFINED BUTTONS

Depending on which menu commands you use often, you may want to create your own user-defined buttons (see Figure 2-2). These buttons allow you to execute a menu command by clicking on the button.

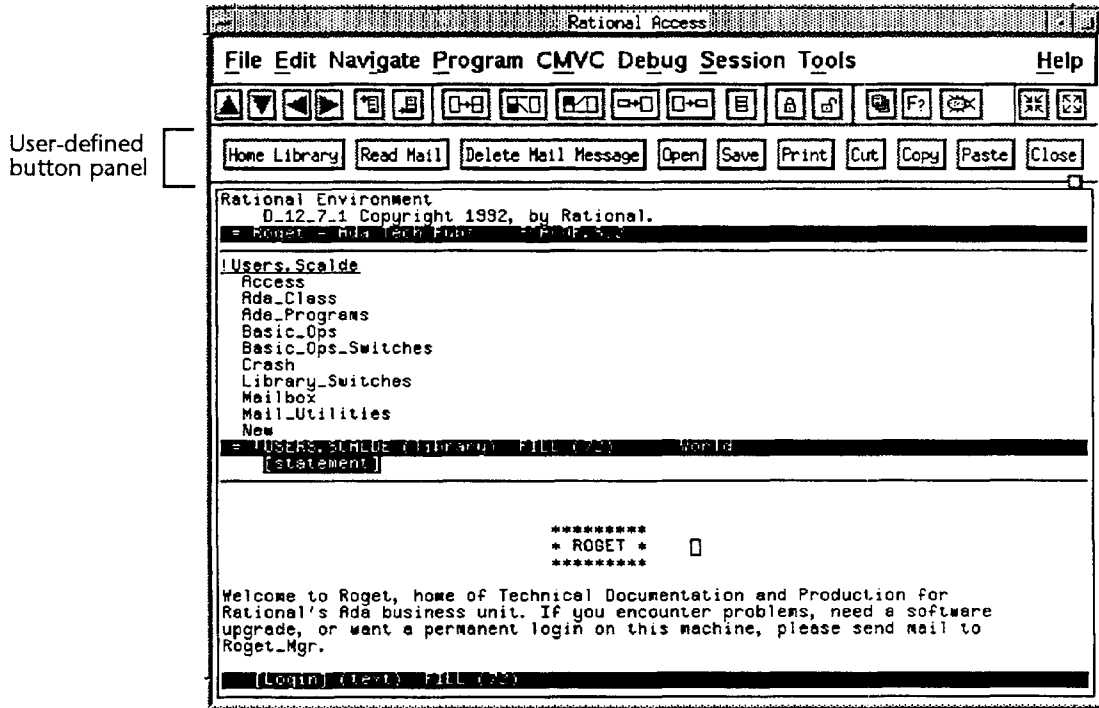


Figure 2-2 The User-Defined Button Panel in the Main Access Window

The user-defined buttons are displayed as the name of the command enclosed in a rectangular border, and they appear directly below the window-control buttons and above the sash. You can create as many buttons as you wish, making space for them by pulling down the sash in the Environment area. The buttons appear left to right, in the order that you create them. Buttons can be saved and deleted between logins, and they are user-specific, not session-specific.

Following are some commands for creating, activating, deleting, and saving user-defined buttons:

- To create a user-defined button, place the pointer on a menu command and press [Control]+click.
- To activate a user-defined button, place the pointer on the button and click.
- To delete a user-defined button, place the pointer on the button and press [Control]+Click.
- To save user-defined buttons, choose Session:Screen:Save Button Panel.

See Chapter 14, "Customizing Your Access Workspace," for more instructions on how to create, execute, and save buttons, and how to change the size of the user-defined button panel.

IMAGE PALETTE

The Rational Access Image Palette provides an updatable listing of the images open under your Environment session. The Image Palette gives you the ability to list all the current images, including those that are not currently displayed in an Environment window, and to recall any of these images to the Environment area.

To display the Image Palette, click the following button on the window-control panel:

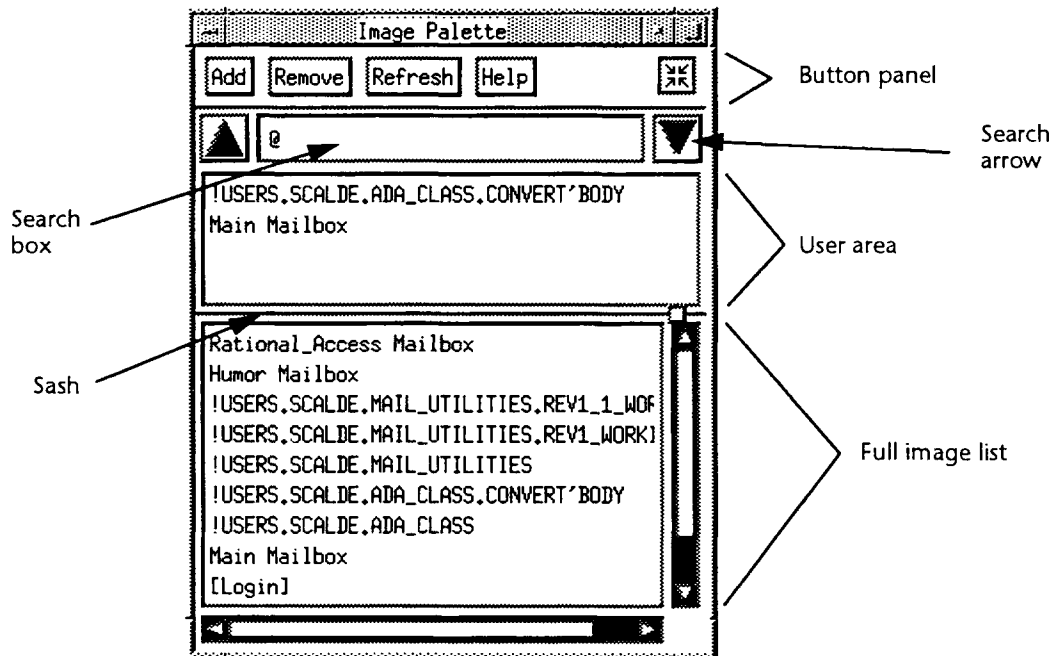


Figure 2-3 *The Image Palette*

The top panel of the palette—the *button panel*—contains buttons that control the palette. The *Refresh* button must be pressed every time you want to update the Image Palette to include new images brought up in the Environment area. (The Image Palette is updated automatically, when you open it.)


The middle panel—the *user area*—contains a user-defined list of Environment windows that can be recalled to the associated main Access window. You can keep a list of important images in the user area by using the *Add* and *Remove* buttons.

The bottom panel—the *full image list*—contains a list of the Environment windows that are on the associated main Access window or have been replaced by other windows but not closed. You can change the relative size of the bottom two panels by placing the pointer on the *sash* control and dragging it up or down.

To search through the full image list, you can use the scroll bars or search for a specific image using the search box and search arrows.

In the Image Palette:

- To redisplay a listed image, double-click the entry in the image list.
- To manually search through the image list, drag the scroll bars.

- To search for a specific image in the list, enter a character string in the search box, using wildcards if desired, and direct the search with the search arrows.
- To add an image from the image list into the user area, click the image name (highlight it), and click the Add button.
- To remove an image from the user area, click the image and click the Remove button.
- To update the Image Palette, click the Refresh button.
- To close the Image Palette, click: 

For more information on the Image Palette, see Chapter 4, "Managing Environment Windows."

FUNCTION KEY PALETTE

The Function Key Palette brings up a list of the commands that are bound to each function key and allows you to execute a function-key command by clicking on its entry in the palette.

To bring up the Function Key Palette, click the following button on the window-control panel: 

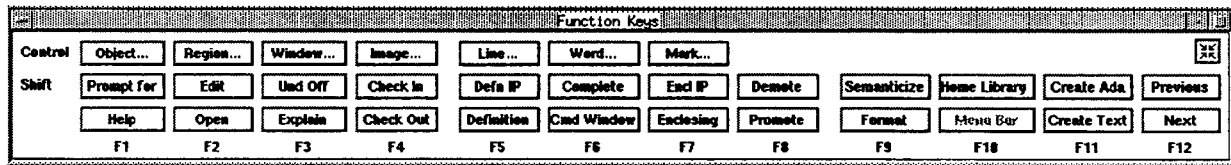



Figure 2-4 The Function Key Palette


In the Function Key Palette:

- To execute a function-key command, click the command on the palette.
- To close the Function Key Palette, click: 

Note: The top seven control buttons on the palette support the item-operation paradigm used in other Rational user interfaces. Clicking on one of these buttons brings up a second-level control palette with object operations buttons.

DEBUGGER PALETTE

The Debugger Palette allows you to debug Ada programs using a persistent control palette.

To display the Debugger Palette, choose Debug:Debugger Commands Palette or click the following button on the window-control panel: 

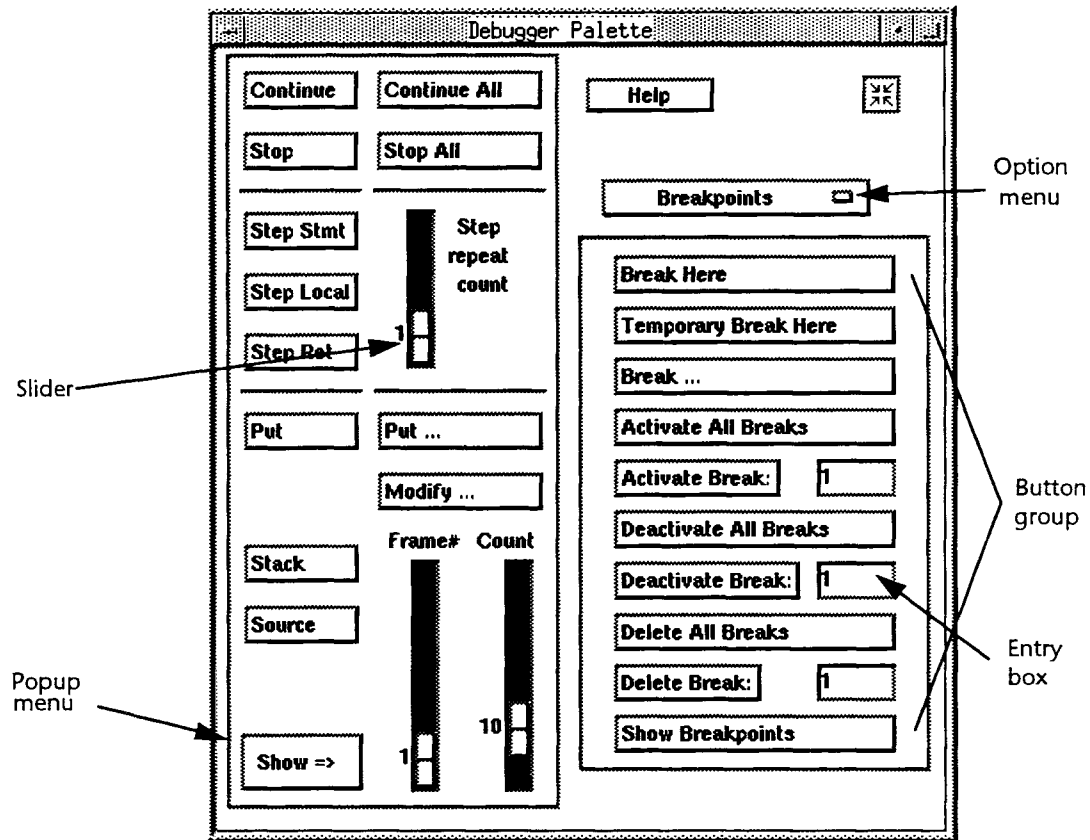



Figure 2-5 *The Debugger Palette*

The top-left section of the Debugger Palette contains controls for executing a program running under the debugger. The bottom-left section of the palette contains controls that pertain to variables. The right side of the palette contains the debugging option menu, which controls sets of buttons for:

- Breakpoints
- Exception Handling
- Task Information
- Task Control
- Debugger Control
- Tracing
- Target
- Machine and Memory
- Quit

In the Debugger Palette:

- To display a set of buttons from the option menu, click the option menu label to reveal the options. Then click the option you want. The new set of buttons replaces the old set.
- To close the palette, click: 

JUST-DO-IT MODE

You can suppress the dialog box for certain commands. This is especially useful if you execute the command with the same dialog-box settings many times in a row, or if you use the default settings. If you want to bypass a dialog box when executing a menu operation, use the Just-Do-It mode, which executes the command without displaying the dialog box.

To execute with Just-Do-It mode, press [Meta] while executing the command. The Environment completes the command using the default selection in the dialog box.

You can change the default settings of a dialog box by setting the new parameters in the box and executing the command or pressing the Cancel button. The next time you execute the command with Just-Do-It mode, it will execute using the new default parameters.

Note that this mode has no effect on most menu commands. If this mode is not implemented for a menu command, pressing [Meta] while clicking brings up the following dialog box:

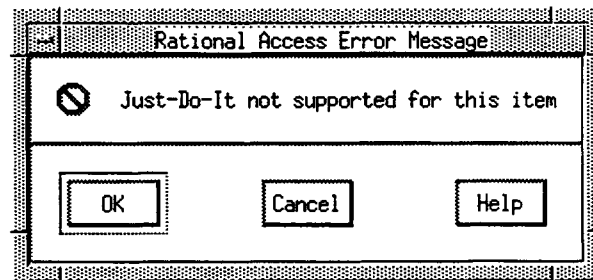


Figure 2-6 *The Error Message for an Unsupported Operation in Just-Do-It Mode*

Press OK or Cancel to exit the error message and continue to the menu command dialog box.

Note that if a second dialog box is required, Just-Do-It mode does not bypass the second box.

The following commands can be executed using Just-Do-It mode:

- File:Print
- CMVC:Accept Changes
- Program:Promote to Coded
- Program:Promote to Installed
- Program:Promote to Source
- Program:Demote to Installed
- Program:Demote to Source
- Program:Demote to Archived

3

Getting Help

This chapter describes how to obtain help and navigate through the various kinds of online help available with Access. Note that there are two help windows:

- The Access help window, which is a separate window from your main Access window
- The Environment help window, which appears in the Environment area of your main Access window

OBTAINING INFORMATION FROM THE ACCESS HELP WINDOW

When you request information about a menu, a dialog box, or a topic on the Help menu, Access generally displays the information in the Access help window. Figure 3-1 shows the help displayed by Help:On Getting Started.

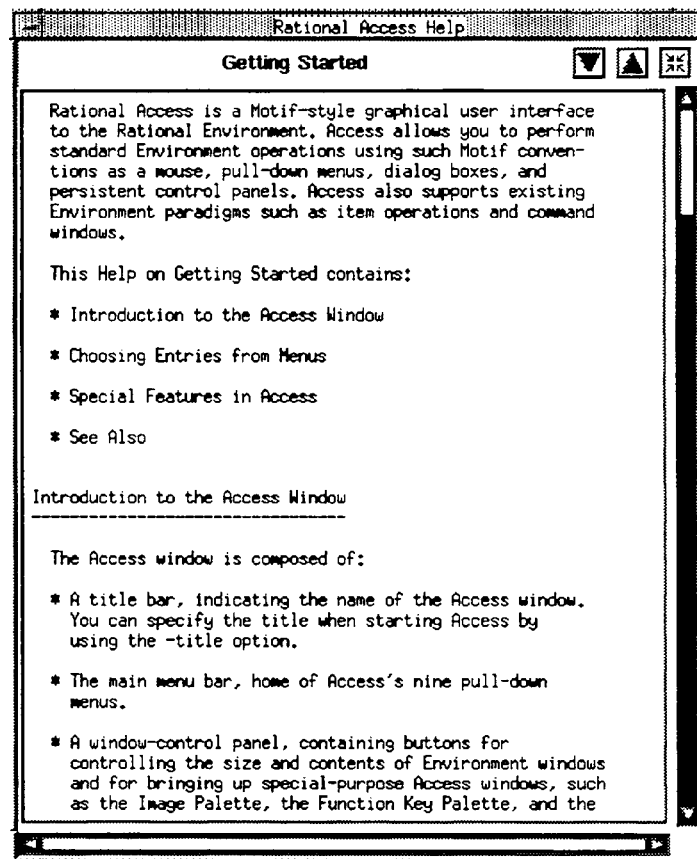



Figure 3-1 The Access Help Window

In the Access help window you can:

- View more text by dragging the slide bars
- Skip to the next heading in the help entry by using the search arrows
- Close the window by clicking: 

GETTING INTRODUCTORY INFORMATION ABOUT ACCESS

Choose Help:On Getting Started.

An Access help window appears with basic Environment and Access information, including an introduction to the Access window, how to choose entries from menus, and special features in Access.

GETTING INFORMATION ABOUT THE ONLINE HELP SYSTEM

Choose Help:On Help.

The Access help window appears with information about how to get help for Environment commands, key and mouse bindings, the window-control button panel, menus, and dialog boxes.

FINDING OUT WHAT A MENU CONTAINS

Choose Help:On Menu:*Desired Menu*.

The Access help window appears with information about the items on the desired menu.

GETTING HELP ON THE WINDOW-CONTROL BUTTON PANEL

1. Choose Help:On Window Panel.

The Access help window appears, and the pointer becomes a question mark when you move it outside the dialog box.

2. Using the question-mark pointer, click on one of the window-control buttons.

A help entry for that button appears in the Access help window.

3. To turn the pointer back into an arrow, click the mouse anywhere in the Environment area.

GETTING HELP ON THE MOUSE

Choose Help:On Mouse.

The Access help window appears with a list of the mouse operations.

GETTING HELP ON KEYS

Getting Help on Key Bindings

Choose Help:On Key Bindings.

The Access help window appears with a list of all standard Access key bindings.

Note: *Site-specific or personal key bindings are not reflected in this text.*

Getting Help on Function Keys

Choose Help:On Function Keys.

The Access help window appears with a list of all function keys and their operations, including all item-operation keys and their operations.

Finding What Command a Key Executes

1. Choose Help:On Key.

A message appears in the message window prompting: Press key to be described.

2. Press the key or key combination to be described.

The Environment help window appears with the name of the command and the key(s) to which it is bound. An explanation from the *Rational Environment Reference Manual* also appears in the window.

FINDING OUT WHAT VERSION OF ACCESS YOU ARE USING

Choose Help:On Version.

A dialog box appears, showing the version of Rational Access that you are currently using.

- To get help with this dialog box from an Access help window, press the Help button.
- To close the Version dialog box, click OK.

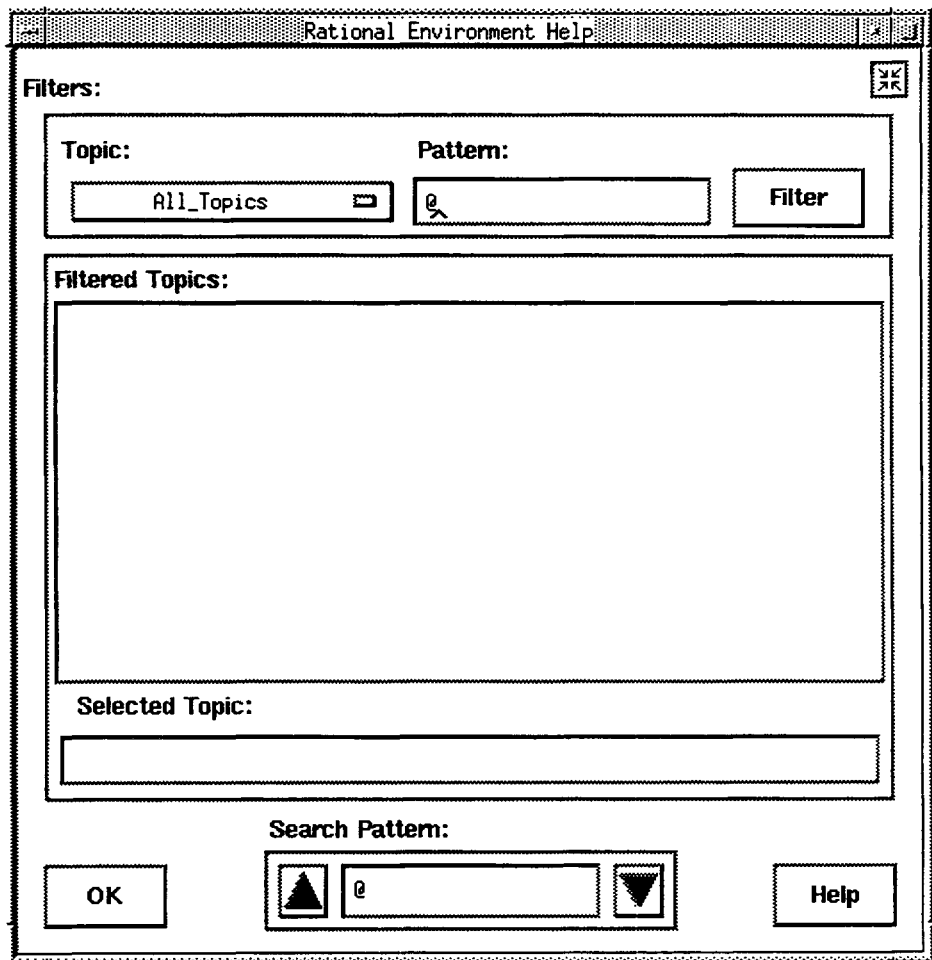
GETTING A LIST OF HELP TOPICS

Getting a List of Environment Commands and Packages

Through the Access help facility, you can obtain information on Environment commands and packages. (Environment commands are Ada subprograms that you usually execute through command windows.) To get a list of help topics:

1. Choose Help:On Environment.

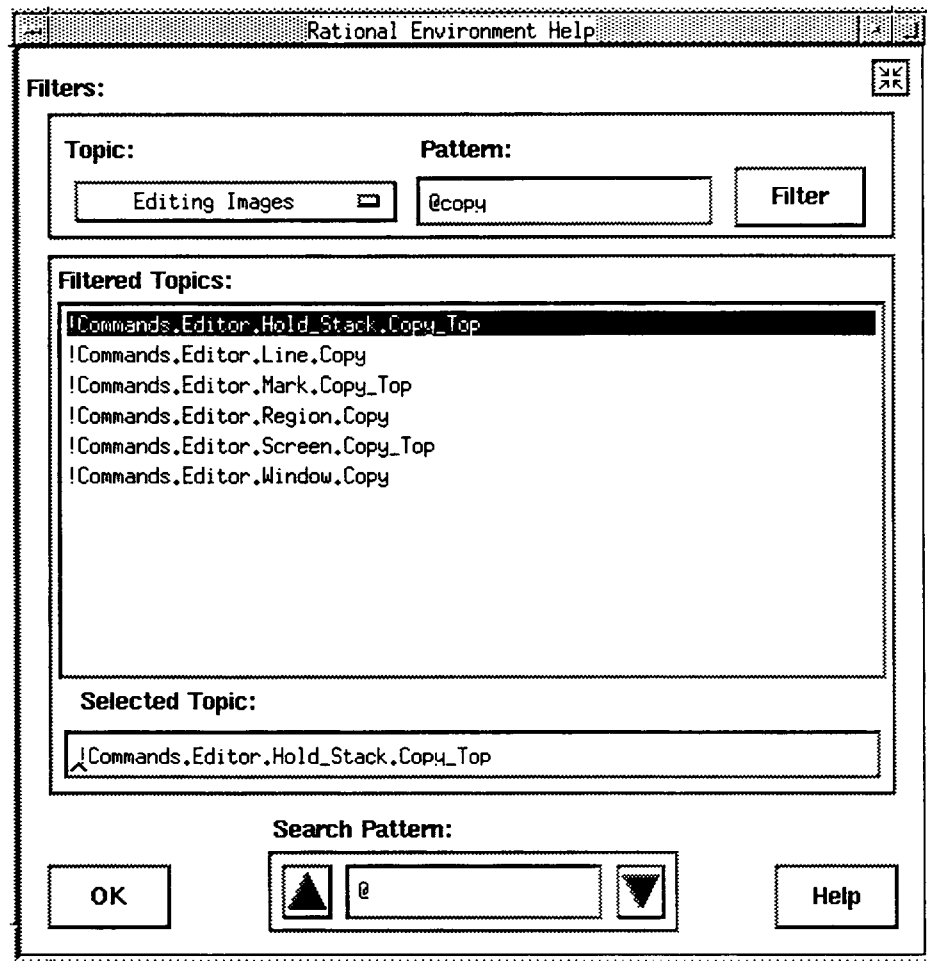
The following dialog box appears:



2. Click the Topic option menu and choose the help topic area (or All_Topics) for which you would like information.
The topic areas are the same as the topics for the books of the *Rational Environment Reference Manual*. To get a list of Access commands, choose #Rational Access from the Topic option menu.
3. Specify a pattern for the list of commands to follow when you filter it.
 - To list all commands for the topic you chose, leave the @ sign in the Pattern entry box.

- To list only certain commands in the list, type an entry (using Environment wildcards for matching library objects) in the **Pattern** entry box.
4. Click the **Filter** button to produce a list of related commands/topics for that help topic area.

For example, choosing **Editing Images** on the **Topic** menu, entering `@copy` in the **Pattern** entry box, and pressing **Filter** displays a list of Environment editing commands that have the word *copy* in them:



Note that the first command in the list is highlighted in reverse video by the location cursor and listed in the **Selected Topic** entry box.

Once the list is displayed, you can:

- Search through the list (see below)
- Scroll through the list by dragging the scroll bar with the mouse
- Select an item by clicking on it with the mouse and click **OK** to get help

The Environment help window appears in the Environment area of your main Access window and displays information about the selected command.

Searching through the List

You can scroll through the filtered list of commands stopping only at a specific command or at commands that match a designated pattern.

1. Enter a specific command name or name fragment in the Search Pattern entry box. Search patterns can include the pattern-matching wildcards shown in Table 3-1.

Table 3-1 Wildcards

Wildcard	Function
?, %	Match any character
{, }	Match beginning and end of line, respectively
[xyz]	Means "The character in this position is x or y or z."
\\	Means "The following character is not a wildcard, even if it looks like one."
*	Means "Zero or more occurrences of the preceding character."
@	Is a synonym for ?* (zero or more occurrences of anything).

Searching is not case-sensitive. An empty search pattern matches nothing.

2. Click the search arrows to search up or down in the filtered list. Every time you click an arrow, the location cursor will stop only at commands that match your search pattern.
3. Click OK to get help for the selected command.

Alternative: Drag the scroll bar to search through the list manually.

Displaying a Subset of Topics in a Filtered List

1. Enter the topic name or fragment (with wildcards) in the Pattern entry box.
2. Press the Filter button.

The filtered list now displays only those commands that match your entry in the Pattern box.

Alternative: Type directly into the Selected Topic box. Wildcards are ignored, but incomplete topics are accepted.

GETTING HELP ON AN ENVIRONMENT COMMAND

If you know the name of the Environment command for which you want help, follow these steps. If not, see above for getting a list of topics.

1. Press [Help].

- A command window appears containing the Environment What.Does command.
2. At the Name parameter, enter the topic, command name, or command-name fragment for the area of interest, and press [Promote].

If only one command exists with that string in its name, information about that command, including a brief description and a list of any keys bound to the command, is displayed in the help window.

If more than one command exists with that string in its name, all the matching commands are listed in the help window. If you want to see the help for one of these items, place the cursor on the line on which the item is located and press [Explain]. The help for that item is then displayed in the help window.

If no commands can be found about that topic, a message appears indicating that no help is available for that topic.

GETTING HELP ON ERRORS

To get additional information about an error in your program or command:

1. Move the cursor onto the underlined error.
2. Choose Help:Explain.

If the Environment has any more information, additional messages about the error appear in the message window.

Alternative: *Instead of choosing Help:Explain, press [Explain].*

DISPLAYING ADA SPECIFICATIONS

To see the Ada specification for an item described in the Environment help window:

1. Place the cursor on the Ada code segment for the item in the Environment help window.
2. Double-click the left mouse button.

The Environment opens a new window containing the Ada specification.

4

Managing Environment Windows

This chapter describes how to manipulate, navigate, and restore Environment windows with Access.

MOVING BETWEEN ENVIRONMENT WINDOWS

1. Place the pointer completely in any part of any Environment window (text or white space).
2. Click the left mouse button.

The Environment cursor appears at the exact location of the pointer.







Note: *If the cursor overlaps two windows, the message window displays the message: The cursor is not in a window.*

Alternative: *To move the Environment cursor to the Environment window above, press [Control][Meta][U] or [Control][Meta][↑]. To move to the Environment window below, press [Control][Meta][N] or [Control][Meta][↓].*

MOVING WITHIN AN ENVIRONMENT WINDOW

Table 4-1 lists the window-control buttons that move your view of the image.

Table 4-1 *Window-Control Buttons for Movement*

Button	Function	Alternative
	Moves your view of the image up toward the beginning of the file	[Shift][↑], [Meta][V], [Control][Z], [Page Up]
	Moves your view of the image down toward the end of the file	[Shift][↓], [Control][V], [Page Down]
	Moves your view of the image to the left	[Shift][←]
	Moves your view of the image to the right	[Shift][→]
	Moves the Environment cursor and your view to the top line of the image.	[Shift][Home], [Shift][Page Up]
	Moves the Environment cursor and your view to the bottom line of the image.	[Shift][End], [Shift][Page Down]

Traversing in a Window Using a Mark

Making a Mark

1. Place the Environment cursor at the location you want to mark.
2. Press [Control][M] or [Control][@].

The location is marked by the Environment, but no change is visible.

Traversing to a Mark

Press [Control][Meta][M].

The cursor returns to the marked location.

Moving the Cursor to the Beginning of the Window Frame

Press [Control][Meta][Home].


Moving the Cursor to the End of the Window Frame

Press [Control][Meta][Home].

CHANGING THE SIZE OF AN ENVIRONMENT WINDOW

Shrinking a Window

1. Begin with the cursor in the window to be shrunk.

2. Click: 

The window shrinks by four lines.

Alternative: Press [Control][.] or [Control][<].

Expanding a Window

1. Begin with the cursor in the window to be expanded.

2. Click: 

The window expands by four lines.

Alternative: Press [Control][.] or [Control][>].

Expanding the Current Window over the Next Frame

1. Place the Environment cursor in the window you want to expand.


2. Click: 

The current window expands to the size of the current window plus the window frame below, replacing any window that might have been in that frame. The window returns to its normal size automatically when the next object is viewed, unless it is locked.

Note: *If your current window is at the bottom of the Environment area, pressing the button shown above expands the window over the window directly above.*

Expanding the Current Window over the Previous Frame

1. Place the Environment cursor in the window you want to expand.

2. Click: 

The current window expands to the size of the current window plus the window frame above, replacing any window that might have been in that frame. Unless it is locked, the window returns to its normal size automatically when the next object is viewed.

Note: *If your current window is at the top of the Environment area, pressing the button shown above expands the window over the window directly below.*


Expanding the Current Window to Full Size

1. Place the Environment cursor in the window to be expanded.

2. Click: 


The current window fills the Environment area (except the message window).

Making All Major Environment Windows the Same Size

Click: 

SPLITTING AN ENVIRONMENT WINDOW

1. Place the Environment cursor in the window to be split.


2. Click: 

The window divides into two windows with identical contents. Each image can be scrolled independently. Making changes to one of these images affects both images.

REMOVING AN ENVIRONMENT WINDOW

Removing a Window Temporarily

1. Place the Environment cursor in the window you want to remove.

2. Click: 

The window disappears, but remains listed in the Image Palette and Environment window directory.

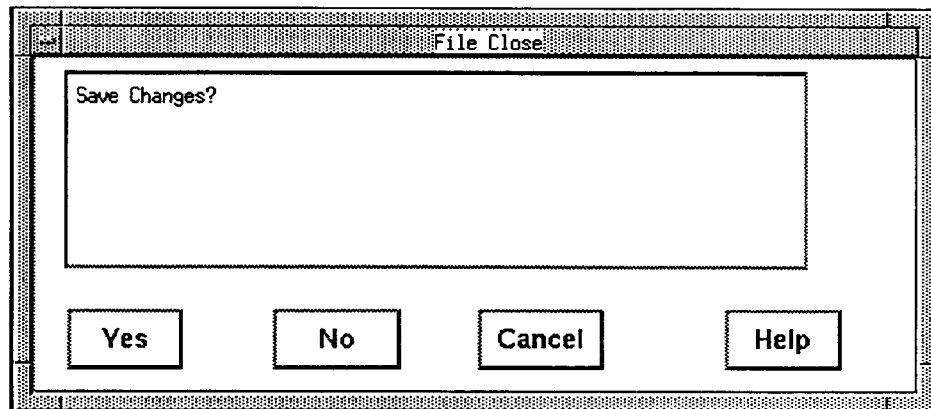
Alternative: Press [Control][Meta][K].

Removing an Image Permanently

1. Place the Environment cursor in the window that contains the image.

2. Choose File:Close.

If changes have been made to the image, the following dialog box appears:



3. Release the image.


- To save the changes and release the image, click Yes.
- To release the image without saving the changes, click No.

The window is no longer listed in the Environment window directory and will disappear from the Image Palette the next time the palette is refreshed.

LOCKING OR UNLOCKING AN ENVIRONMENT WINDOW

Locking a Window

1. Place the Environment cursor in the window to be locked.


2. Click: 

An *at* sign (@) appears in the window banner.

This window will not be removed unless you explicitly remove or unlock it.

Alternative: Press [Control][Meta][P] until the *at* sign (@) appears in the window banner.

Unlocking a Window

1. Place the Environment cursor in the window to be unlocked.
2. Click: 

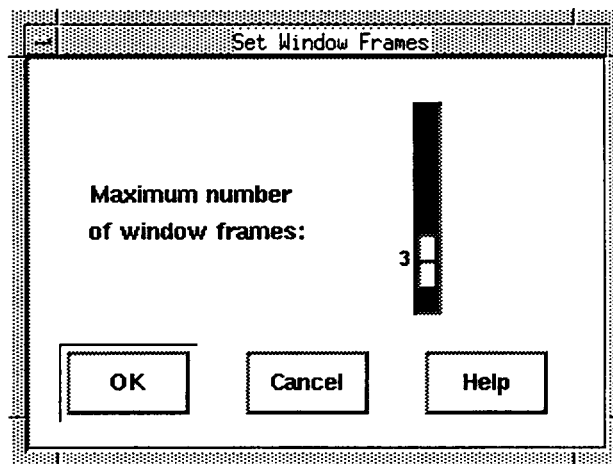
The *at* sign (@) disappears from the window banner.

Alternative: Press [Control][Meta][E] to unlock the window.

SETTING THE NUMBER OF WINDOW FRAMES

1. Choose Session:Screen:Window Frames.

The following dialog box appears:



2. Drag the slider to set the maximum number of window frames you want to appear in the Environment area. The default is 3, and the maximum is 16.
3. Click OK to implement the change.

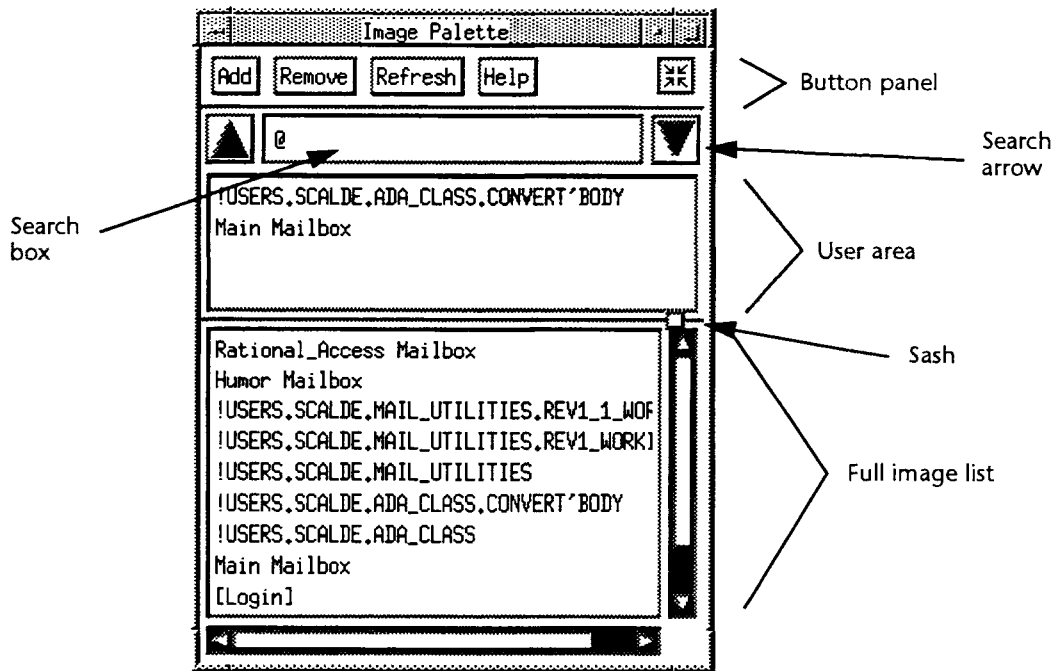
Note: You can have more than the maximum number of windows by using the Copy Window button.

GETTING A LIST OF ENVIRONMENT WINDOWS

The Rational Access Image Palette provides an updatable listing of the images open under your Environment session. The Image Palette gives you the ability to list all the current images, including those that are not currently displayed in an Environment window, and to recall any of these images to the Environment area.

To display the Image Palette, click:





- The top panel of the palette contains buttons that control the palette.
- The middle panel contains a user-defined list of Environment windows that can be recalled to the associated main Access window.
- The bottom panel contains a complete list of the Environment windows that are on the associated main Access window or have been replaced by other windows but not closed.

Note: If the pathnames listed in the Image Palette extend beyond the visible window area, or if the list extends below the visible window area, scroll bars will appear on the right and bottom of the Image Palette.

Redisplaying a Window from the Image Palette

Double-click the entry for the window you want to redisplay.

The window reappears in the Environment area of the associated main Access window.

Searching for a Window

1. Enter the full name of the image, or a partial name using wildcards, in the search box in the Image Palette.
2. Click the search arrows to search forward or backward in the list for the string you entered.
The location cursor in the Image Palette will move to the first occurrence of the string.
3. Click the arrow again to continue the search.

Alternative: You can search through the image list manually by dragging the scroll bar with the mouse.

Updating the Image Palette

Once the Image Palette has been displayed, the image list must be updated by you to reflect changes made in the Environment area.

Click the **Refresh** button.

The image list is updated.

Note: If you close the Image Palette and then reopen it, the image list is updated automatically.


Setting Up a Standard Set of Windows

1. Click the entry of the window you want to save in the Image Palette. This places the entry in reverse video.
2. Click the **Add** button in the Image Palette.
The entry appears in the user area of the Image Palette and will remain until you log out, even if you close the Image Palette.

Changing the Size of the User Area and Full Image List

Place the pointer on the sash control and drag it up or down to change the size of the middle panel and bottom panel.

Closing the Image Palette

To close the Image Palette, click: 

Finding Windows with Uncommitted Changes

To find which windows have uncommitted changes, use the Environment window directory.

Press **[Control][/]** or **[Control][?]** to display the window directory.

The window directory is displayed in a new Environment window. Objects that are closed for editing are marked by an equals sign (=). Objects that have unsaved changes are marked by an asterisk (*).

SAVING AND RESTORING A SET OF WINDOWS

You can save your current set of windows and recall it at any time.

Saving a Set of Windows

Choose **Session:Screen:Screen Push**.

The Environment saves the current configuration of windows.

Restoring a Set of Windows

Choose **Session:Screen:Screen Pop**.

The last saved set of windows appears in the Environment area of your Access window.

5

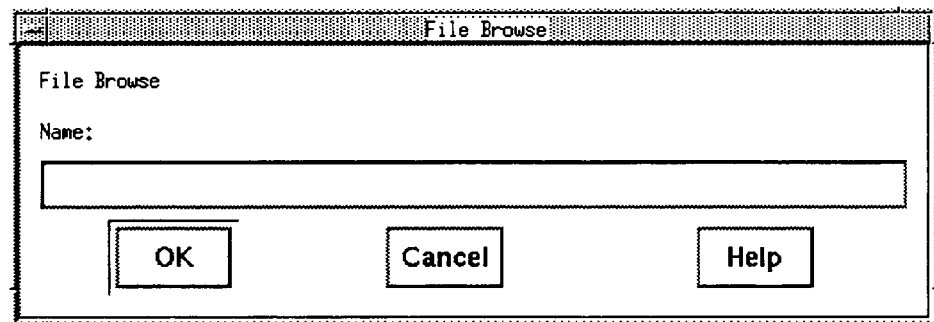
Traversing the Environment

This chapter describes how to traverse the Environment library hierarchy.

VIEWING ANY OBJECT

1. Choose File:Browse.

The following dialog box appears:



2. Enter the name of the object you want to view in the Name entry box.
3. Click OK.

The specified object is displayed in an Environment window.

Note: The standard wildcards for matching library objects are accepted.

VIEWING AN OBJECT IN THE CURRENT CONTEXT

Making the Object Appear in the Next Window to Be Replaced

Place the Environment cursor on the name of the object to be viewed and double-click the left mouse button.

The object appears in a window.

Alternative: Press [Definition] or execute the Navigate:Definition menu command.

Making the Object Appear in the Same Window

Place the Environment cursor on the name of the object to be viewed and [Shift]+double-click the left mouse button.

The new object replaces the object in the current window.

Alternative: Press [Definition in Place] or [Control][Meta][→].

VIEWING THE ENCLOSING LIBRARY

Making the Library Appear in the Next Window to Be Replaced

Double-click the right mouse button.

The enclosing library appears in a window.

Alternative: Press [Enclosing] or execute the `Navigate:Enclosing` menu command.

Making the Library Appear in the Same Window

Press [Shift] and double-click the right mouse button.

The enclosing library replaces the object in the current window.

Alternative: Press [Enclosing in Place] or [Control][Meta][←].

VIEWING YOUR HOME LIBRARY

Choose `Navigate:Home Library`.

Your home library appears in a window.

Alternative: Press [Home Library].

6

Browsing Ada Programs

This chapter describes how to traverse through specifications and bodies of Ada units and packages.

MOVING BETWEEN THE SPECIFICATION AND BODY OF AN ADA UNIT

Choose `Navigate:Other Part`.

If you are in an Ada specification, its body appears in a window. If you are in a body, its specification appears in a window.

VIEWING A UNIT'S PARENT

Double-click the right mouse button.

If the Environment cursor is in:

- A subunit, the parent unit body appears in an Environment window
- A unit, the enclosing library appears in an Environment window

Alternative: [Shift]+double-click the right mouse button or press [Control] [Meta] [←].

SHOWING OCCURRENCES OF A DEFINED ADA NAME

Showing References to a Defined Ada Name

1. Begin in the window containing the Ada name of interest.
2. Select an occurrence of the Ada name.
3. Choose `Program:Show Usage`.

References to the Ada name within the current unit are underlined. To step through, choose `Navigate:Next Item` to move to the next underlined item or `Navigate:Previous Item` to move to the previous underlined item.

References to the Ada name in other units are listed in a separate window.
4. Select a unit.
5. Double-click the left mouse button to view the unit with the using occurrence.

A window appears displaying the selected unit with all occurrences of the Ada name of interest underlined.
6. Use `Navigate:Next Item` or `Navigate:Previous Item` to step through.

Note: *The default limit of this operation is immediate dependents, and the default scope is all worlds.*

Alternative: *Press [Next Item], [Meta][N] or [Meta][↓] to move the Environment cursor to the next occurrence; press [Previous Item], [Meta][U], or [Meta][↑] to step back to the previous occurrence.*

Showing Unused Ada Constructs

The Ada unit must be in the installed or coded state.

Choose Program:Show Unused.

Unused occurrences of all Ada constructs are underlined.

GETTING THE DEFINITION OF AN IDENTIFIER

Place the pointer on the identifier and double-click the left mouse button.

A window containing the definition of the declaration appears.

VIEWING THE SPECIFICATION OF AN ENVIRONMENT PACKAGE

Here is a convenient shortcut for displaying the specifications of Ada units provided as part of the Environment (for example, for viewing the specification of package Compilation, which contains the compilation commands).

1. Press [Prompt For].
A Prompt For message appears in the message window.
2. Press [Definition].
3. Enter the simple name of the Ada unit at the prompt for the Name parameter preceded by the backslash (\) character (for example, \Compilation).
The \ character causes the Environment to use your searchlist when looking for the name following it. For information about searchlists, see package Search_List in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual*.
4. Press [Promote] to execute the command.

Note: *This shortcut for viewing Environment package specifications works for most Environment packages. If the shortcut fails, an error message appears, and you have to traverse to the specification instead.*

7

Writing Ada Programs

In the Environment, each library-level Ada unit is stored as a separate library object of class Ada. Environment Ada units differ from text files in that they have an underlying structured representation (called DIANA).

Compiling a program in the Environment entails *promoting* all of its units through a series of *state changes*, each of which adds information to the unit's DIANA representation. Each unit state represents a distinct phase in compilation:

- Units that are not of current interest can be stored in the *archived state*.
- Entering and parsing code is done interactively in the *source state*.
- Integration between units occurs when they are promoted to the *installed state*.
- Executable code is generated when units are promoted to the *coded state*.

These states are ordered from low to high; a unit must be promoted from source (or archived) through installed to coded before it can execute. Thus, multiple states replace the notion of multiple files; no additional library objects are created to contain object code or executable images. (Note that if you are using the R1000 code generator, linking and loading are done dynamically as part of execution.)

Demoting units is the counterpart to promoting them. Whereas promoting a unit adds information to the unit's DIANA representation, demoting a unit releases information from the DIANA representation. Most often you demote units so that you can edit them. Units also may need to be demoted to maintain semantic consistency and to allow the demotion of units on which they depend. Note that many changes can be made to units that are installed or coded without demoting them.

This chapter describes how to create, edit, and compile Ada units using Access. For an introduction to writing Ada programs, see the *Rational Environment User's Guide*. For complete information about the Environment's compilation model and the Environment commands available for compilation, see the Library Management (LM) book of the *Rational Environment Reference Manual*.

CREATING AN ADA UNIT

Creating a New Ada Unit

1. Place the Environment cursor in the library in which you want to create a new Ada unit.
Ada units can be created in any library; however, in subsystem views, units generally are created in the Units directory or one of its subdirectories.
2. Choose File:New:Ada.

A window is created for the new Ada unit. The window displays a `comp_unit` prompt in which you can enter the Ada specification or body. The library now contains a temporary name of the form `_Ada_#_` where `#` is some number. The temporary name is replaced with the simple name of the unit the first time it is promoted to the installed state.

Note that each Ada compilation unit in your program must be created individually. This is necessary because each Ada unit in the Environment is a distinct object, whereas file-based development systems allow multiple units (for example, a specification and a body) to be put into a single file.

Alternative: Press [Create Ada] to create a new Ada unit.

Creating a Body from a Specification Automatically

You can use the steps described in "Creating a New Ada Unit," above, to create both the specification and the body. However, you can also use the Environment's automated facility for generating a unit body from its specification:

1. Place the Environment cursor in the unit specification.
2. Choose **Program:Build:Build Body**.

The Environment uses the declarations in the unit specification to generate a skeletal package body containing a template for each visible subprogram. The `[statement]` prompts indicate where statements need to be filled in. The unit body is in the source state and is open for editing.

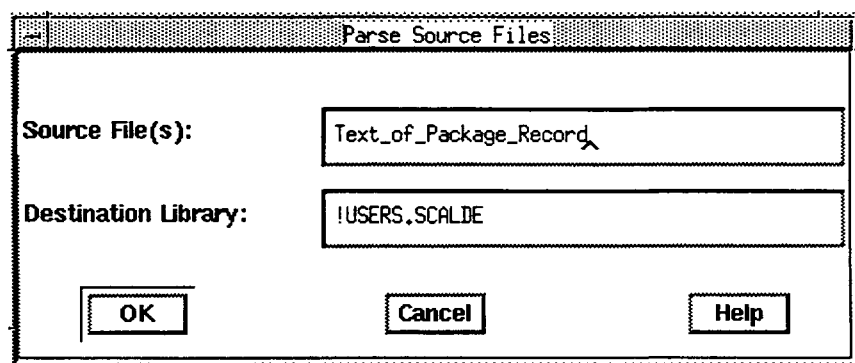
Building a Unit from a Text File

Because Environment Ada units differ from text files in that they have an underlying structured representation (called DIANA), you usually create and edit Ada source code directly in Ada units.

Occasionally, however, Ada source code may already exist in a text file. For example, the Ada source code may have been created on another host and need to be transported to the R1000. In this case, you will need to *parse* the source file into an Ada unit. To do so:

1. Place the Environment cursor in the text file.
2. Choose **Program:Build:Parse Source Files**.

The following dialog box appears:



3. Enter the name of the file(s) to be parsed in the **Source File(s)** entry box.
4. Enter the name of the destination library in the **Destination Library** entry box.
5. Click **OK**.

The newly created objects are characteristic of Ada units in the Environment and are created in the source state. The original text file containing the Ada source remains intact and unchanged.

Creating a Subunit

1. Place the Environment cursor in the Ada unit body that is to be the parent of the subunit.
2. Enter the Ada subunit stub notation. You might enter, for example, `procedure foo is separate;`
3. Press **[Format]**.
4. Select the stub.
5. Press **[Edit]** to edit the selected stub.

A new window containing the skeletal subunit appears. The name of the subunit appears in the library under the parent unit.

PROMOTING ADA UNITS

Compiling a program in the Environment consists of promoting all of its units from the source state (or the archived state) through the installed state to the coded state, at which point the unit can be executed. You can promote units to either the next higher unit state or directly to a specific unit state.

Note that to ensure semantic consistency, the Ada units that compose a program must be promoted in the order specified by the *Reference Manual for the Ada Programming Language*. In general:

- A unit specification must be promoted before any units that depend on (*with*) it.
- A unit specification must be promoted before its corresponding body.
- A unit body must be promoted before its subunits. (Note that when promoting to the coded state, a package subunit, generic package subunit, or task subunit must be promoted *before* its parent body.)

The Environment maintains databases of dependencies between units that allow it to compute the compilation order. If an attempted state change would violate compilation-order rules, the operation is rejected and the state change does not take place. Furthermore, Access commands determine the additional units that need to be promoted to maintain semantic consistency and usually attempt to promote them accordingly.

For more information about compilation order and promoting Ada units, see the Library Management (LM) book of the *Rational Environment Reference Manual*.

Promoting an Ada Unit to the Next Unit State

You can promote an Ada unit to the next higher unit state (for example, from source to installed) without bringing up a dialog box or setting parameters.

1. Place the Environment cursor in the Ada unit to be promoted.
2. Choose Program:Promote.

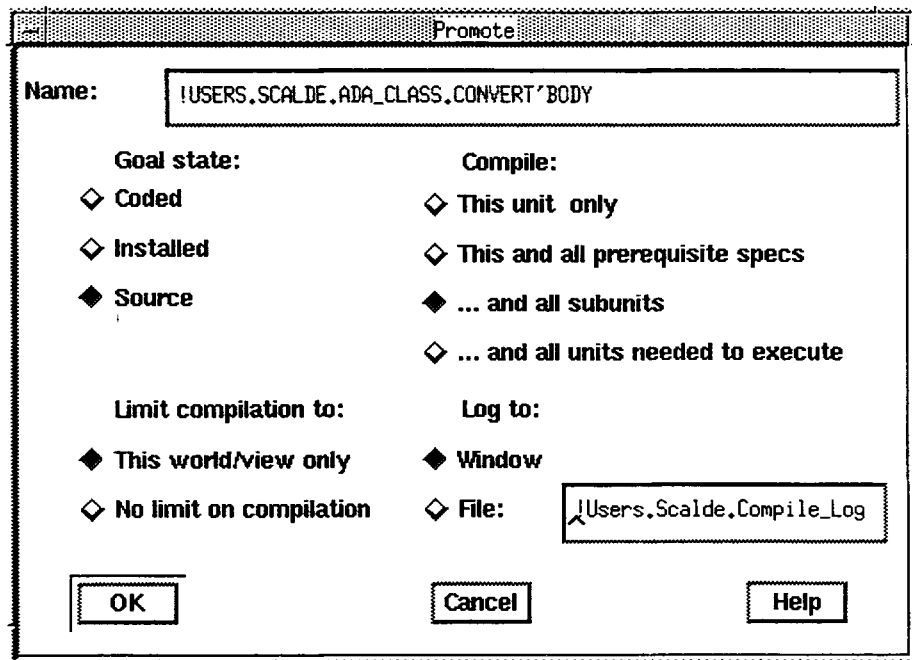
The Ada unit is promoted, and a message appears in the message window describing the procedure. Note that this operation promotes only the specified unit and, if the specified unit is a body, its corresponding specification.

Alternative: Press [Promote] instead of choosing Program:Promote.

Promoting Ada Units to a Specific State

1. Place the Environment cursor in the Ada unit (or in one of multiple units) to be promoted. To promote all the units in a library, place the Environment cursor on the name of the library.
2. Choose the desired compilation level:
 - Program:Promote to Source
 - Program:Promote to Installed
 - Program:Promote to Coded

The following dialog box appears:



3. Ensure that the units you want to promote are specified in the Name entry box:
 - To promote a single unit, specify the name of that unit.
 - To promote all the units in a library, specify the name of that library.

- To promote multiple units, specify an appropriate naming expression using set notation ([unit1, unit2]) or the standard wildcard characters for specifying pathnames:
 - # matches any single character
 - @ matches zero or more characters
 - ? matches zero or more nonworld name components
 - ?? matches zero or more name components, including worlds
4. Verify that the compilation level you chose is reflected in the Goal state field.
 5. In the Compile field, choose the set of related units that are to be promoted automatically:
 - To compile only the named unit (with no prerequisites), select This unit only.
 - To compile the named unit and any prerequisite specs needed for compilation, select This and all prerequisite specs.
 - To compile the named unit and its subunits (including all prerequisite specs), select ...and all subunits.
 - To compile the named unit and all subunits and prerequisite bodies and specs needed for compilation, select ...and all units needed to execute.
 6. In the Limit compilation to field, choose the area in the library hierarchy in which the command can operate:
 - To limit compilation to only objects in the current world or subsystem view, select This world/view only.
 - To allow compilation to include everything in the closure, select No limit on compilation. Note that selecting this option may slow compilation.
 7. Choose where to write the log of the procedure in the Log to field:
 - To display the log of the compilation in an I/O window, select Window.
 - To route the log to a file, select File. The Environment will create a default text file, or you can edit the filename in the File entry box.
 8. Click OK.

DEMOTING ADA UNITS

Demoting units is the counterpart to promoting them. Whereas promoting a unit raises it to a higher state, demoting a unit changes the state of the unit to a lower state. For example, changing a unit from the coded to the installed state entails demoting the unit. There are two main reasons for demoting a unit:

- To edit the unit. Demoting a unit to the installed state allows you to edit it using incremental operations. Demoting a unit to the source state allows you to arbitrarily edit it using basic text-editing operations.
- To allow the demotion or deletion of another unit (specifically, a unit *withed* by the specified unit), thus preserving semantic consistency and maintaining the compilation order.

As when promoting units, the Ada units that compose a program must be demoted in the order specified by the *Reference Manual for the Ada Programming Language*. This is the exact opposite of the order summarized in “Promoting Ada Units,” above.

For more information about compilation order and demoting Ada units, see the Library Management (LM) book of the *Rational Environment Reference Manual*.

Demoting to the Previous Unit State

You can demote an Ada unit one level (for example, from installed to source), without bringing up a dialog box or setting parameters.

1. Place the Environment cursor in the Ada unit to be demoted.
2. Choose Program:Demote.

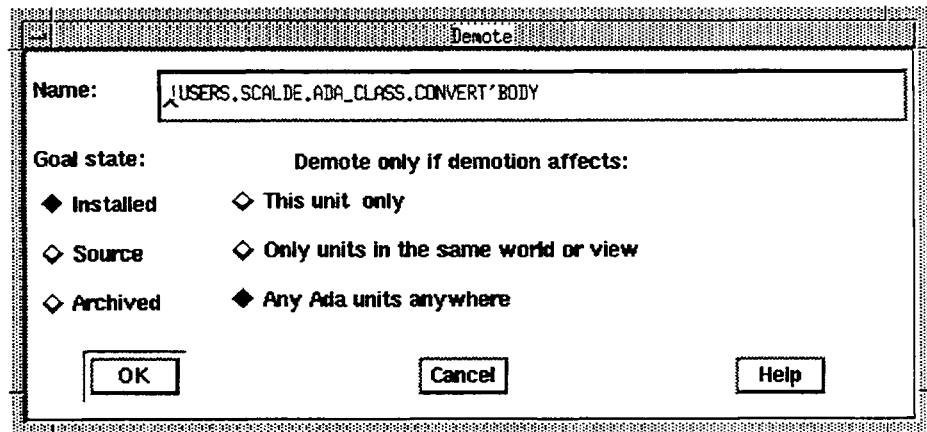
The Ada unit is demoted, and a message appears in the message window describing the procedure.

Alternative: Press [Demote] instead of choosing Program:Demote.

Demoting Units to a Specific State

1. Place the Environment cursor in the Ada unit to be demoted.
To demote all the units in a library, place the Environment cursor on the name of the library to be demoted.
2. Choose the desired compilation level:
 - Program:Demote to Installed
 - Program:Demote to Source
 - Program:Demote to Archived

The following dialog box appears:



3. Ensure that the units you want to demote are specified in the Name entry box:
 - To demote a single unit, specify the name of that unit.
 - To demote all the units in a library, specify the name of that library.
 - To demote multiple units, specify an appropriate naming expression using set notation ([unit1, unit2]) or the standard wildcard characters for specifying pathnames:
 - # matches any single character
 - @ matches zero or more characters
 - ? matches zero or more nonworld name components
 - ?? matches zero or more name components, including worlds
4. Verify that the compilation level you chose is reflected in the Goal state field.

5. Choose the desired option in the **Demote only if demotion affects** field.
 - To demote only the named Ada unit, click **This unit only**.
 - To demote only the named Ada unit and its prerequisite units in your current world or subsystem view, click **Only units in the same world or view**.
 - To demote the named Ada unit and all of its prerequisite units anywhere, click **Any Ada units anywhere**.

Caution: *This may cause massive demotion.*

6. Click OK.

A log of the procedure appears in an I/O window.

Note: *Once an Ada unit is demoted to archived state, it must be promoted to source before you can view or modify it.*

SELECTING PARENT/CHILD ITEMS IN AN ADA UNIT

Selecting the parent selects successively larger items in an Ada unit, and selecting the child selects successively smaller items, based on the following hierarchy:

- Identifier that the Environment cursor is on
- Assignment that includes that identifier
- Statement
- Block
- Subprogram within the unit
- Entire unit

Note: *These operations work differently in a text file. See “Selecting the Parent or Child” in Chapter 10, “Editing Text.”*

MODIFYING UNITS

You can make changes to existing units in any state; however, your changes will affect the unit based on its compilation level. For major changes, you may need to demote the unit.

- **Source state:** Because semantic dependencies are not established by or to units in the source state, units in the source state can be edited, copied, moved, or deleted without making related units obsolete.
- **Installed state:** Installed units cannot be edited arbitrarily, but they can be modified using incremental operations, which check for dependencies to prevent invalidation of the program. On units in the installed state you can:
 - Insert new pragmas, statements, upwardly compatible declarations, and stand-alone comments (comments on a separate line)
 - Delete or change existing pragmas, statements, declarations with no dependents, and stand-alone comments
- **Coded state:** Coded units also cannot be edited arbitrarily, although some changes are allowed using incremental operations. The incremental operations that can be performed on coded units are determined by the target for which the units are being compiled. For the R1000 target, the incremental operations

allowed on coded units are the same as those listed for installed units, with two restrictions:

- Incremental operations on statements are not allowed in coded specifications.
- Incremental operations are not allowed in coded bodies, except on comments.

Adding to a Unit

In the Source State

1. Place the Environment cursor in the unit.
2. If the Ada unit is still in read-only mode, choose File:Open to open the file.
3. Go to the position where the new statement, declaration, or comment is to be added.
4. Enter the changes using basic editing operations.

Alternative: Press [Open] instead of choosing File:Open.

In the Installed or Coded State

Adding a Declaration or Statement

1. Place the Environment cursor in the unit.
2. If the unit is a body and is in the coded state, choose Program:Demote to Installed. (You do not need to demote the unit if it is a specification.)
3. Place the Environment cursor at the location that you want to insert the new statement or declaration.
4. Choose Program:Program Incremental:Incremental Insert.
A new window appears with the banner labeled either statement or declaration, depending on the location of the insertion point.
5. Enter the new statement or declaration.
Note that multiple statements or declarations can be entered per insertion point.
6. Choose Program:Promote to Coded to return the unit to the coded state, if so desired.

The new window disappears, and the prompt in the unit is replaced by the actual statement or declaration.

Note: If there are any errors, the promotion will fail. Errors will be underlined in the unit.

Adding a Comment

1. Place the Environment cursor at the location that you want to insert the new comment.
2. Choose Program:Program Incremental:Incremental Insert.
A new window appears with the banner labeled either statement or declaration, depending on the location of the insertion point.
3. Enter the new comment.
Note that multiple comments can be entered per insertion point.
4. Format and semanticize.
5. Correct any errors.
6. Choose Program:Promote to Coded to return the unit to the coded state, if so desired.

The new window disappears, and the prompt in the unit is replaced by the actual comment.

Incrementally Changing an Existing Unit

In the Source State

1. Place the Environment cursor in the unit.
2. If the Ada unit is still in read-only mode, choose **File:Open** to open the file.
3. Go to the position where the statement, declaration, or comment is to be changed.
4. Enter the changes using basic editing operations.
5. Format and semanticize.
6. Correct any errors.

In the Installed or Coded State

1. Place the Environment cursor in the unit.
2. If the Ada unit is not a package specification and is already coded, choose **Program:Demote to Installed** to demote the unit to the installed state.
 Note that if the change you want to make consists only of Ada comments, you do not need to demote the unit.
3. Go to the end of the statement, declaration, or comment to be changed.
4. Select the entire statement, declaration, or comment.
5. Choose **Program:Incremental:Incremental Edit**.
 The selected statement, declaration, or comment becomes a prompt, and a window with the statement, declaration, or comment appears on the screen.
 Note that if the selected declaration has dependents, the edit operation will not succeed until all dependents are demoted to source.
6. Enter the changes.
 Note that multiple declarations, statements, or comments can be entered.
7. Choose **Program:Incremental:Incremental Promote** to promote the changes.

Note: If there are any errors, the promotion will fail. Errors will be underlined in the unit.

Changing Code into a Comment

1. Select the string to be made into a comment.
2. Choose **Edit:Make into Comment**.

Two dashes will appear before the string, signifying a comment.

Changing a Comment into Code

1. Select the region to be uncommented.
2. Choose **Edit:Uncomment**.

The two dashes before the region will disappear.

Deleting Part of an Existing Unit

In the Source State

1. Place the Environment cursor in the unit.
2. If the Ada unit is still in the read-only mode, choose **File:Open** to open the unit.
3. Go to the position where the statement, declaration, or comment is to be deleted.
4. Select the statement, declaration, or comment.
5. Choose **Edit:Cut**.

Alternative: Press [Open] instead of choosing **File:Open**.

In the Installed or Coded State

1. Place the Environment cursor in the unit.
2. If the Ada unit is not a package specification and is already coded, choose **Program:Demote to Installed** to demote the unit to the installed state.
Note that if the deletion you want to make consists only of Ada comments, you do not need to demote the unit.
3. Select the statement, declaration, or comment.
4. Choose **Program:Incremental:Delete** to delete the statement, declaration, or comment.

The selected statement, declaration, or comment is removed.

Note: All dependents must be in source state. If not, the Environment will display a list of the units and constructs in those units that must also be demoted. You can:

- Select units in the list, press [Complete] to show indirect dependencies, and then demote them.
- Go to units in the list and incrementally demote just the dependent constructs.

Changing the Name or Kind of an Ada Unit

In the Source State

1. Place the Environment cursor on the library name of the Ada unit to be changed.
2. Move the cursor to the line containing the Ada unit.
3. Choose **Program:Build:Withdraw**.
The selected Ada unit is replaced by a temporary name, and a window with the Ada unit appears on the screen. The unit can be edited.
4. Change the unit name, parameter profile, or unit kind.

In the Installed or Coded State

1. Place the Environment cursor in the library containing the Ada unit to be changed.
2. Select the Ada unit.
3. Choose **Program:Build:Withdraw**.
The selected Ada unit is replaced by a temporary name, and a window with the Ada unit appears on the screen. The unit is in the source state.

Note that if the selected unit has dependents, the withdraw operation will not succeed until all dependents are demoted to source.

4. Change the unit name, parameter profile, or unit kind.
5. Choose **Program:Promote:Desired Compilation Level** to promote the unit.

The temporary name in the library is replaced by the new actual name for the unit.

Note: *If there are any errors, the promotion will fail. Errors will be underlined in the unit.*

ADDING A SUBPROGRAM TO A PACKAGE

These procedures assume that the subprogram is to be added to both the specification and the body of the package.

In the Source State

1. Place the Environment cursor at the location in the package where the new subprogram specification is to be added.
2. Choose **File:Open** to open the specification of the Ada package.
3. Enter the new subprogram specification.
4. Format and semanticize.
5. Correct any errors.
6. Promote the unit specification to the installed state using **Program:Promote to Installed**.
7. Select the subprogram specification.
8. Choose **Program:Build:Build Body** to create the subprogram body.

The skeletal subprogram body is placed at the end of the existing package body. The subprogram body may contain prompts where you need to enter code.

9. Go to the subprogram body and add any necessary code.
10. Promote the package body to installed.

Note: *If there are any errors, the promotion will fail. Errors will be underlined in the unit.*

Alternative: Press [Open] instead of choosing **File:Open**.

In the Installed or Coded State

1. Place the Environment cursor at the location in the package where the new subprogram specification is to be added.
2. Choose **Program:Incremental:Incremental Insert**.

A new window with a declaration prompt is created for editing. A temporary name appears in the library under the package specification to which you are adding the subprogram.

3. Enter the new subprogram specification at the prompt.

Note that multiple subprogram specifications can be entered per insertion point.

4. Choose **Program:Incremental:Incremental Promote** to promote the declaration.

5. Select the subprogram specification.
6. Choose **Program:Build:Build Body** to create the body.
7. Enter the subprogram.
8. Choose **Program:Promote to Installed** or **Program:Promote to Coded** to promote the subprogram body.

The window is replaced by a window displaying the existing package body with the new subprogram installed.

***Note:** If there are any errors, the promotion will fail. Errors will be underlined in the unit.*

MAKING A PACKAGE OR SUBPROGRAM BODY INTO A SUBUNIT

1. Select the region you want to make into a subunit.
The unit must be in the source or installed state.
2. Choose **Program:Build:Make Separate**.

A new window with the subunit appears and the parent unit has an appropriate subunit stub. Note that the subunit is now in the source state.

MAKING A SUBUNIT IN-LINE IN THE PARENT UNIT

This operation changes the subunit in the image or selected subunit stub from a subunit to an in-line program unit.

1. Place the Environment cursor in the parent Ada unit in either the source or the installed state.
2. Choose **Program:Build:Make In-Line**.

The subunit stub is replaced by the actual subunit code. Note that the in-line unit is in the same state as the parent.

SAVING INCOMPLETE UNITS

The unit must be in the source state.

1. Place the Environment cursor in the unit.
2. Choose **File:Save**.

The source code is saved.

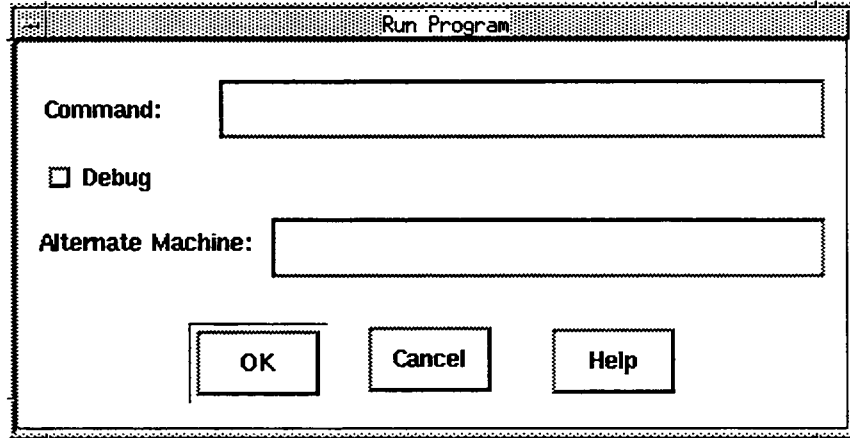
***Note:** Unless the unit has already been installed once, its name will remain in the form `_Ada_#`, where # is some integer.*

EXECUTING A LIBRARY-LEVEL PROGRAM

1. Select the program to be executed.
2. Choose **File:Run**.

The program executes. If the program needs parameters filled in, the Environment opens a command window and prompts for parameters.

Note: *If the program to be executed is not coded, File:Run codes it. If the Environment cursor is on a program that can be executed but it is not selected, a message appears in the message window informing you to select something. If the cursor is not on something that can be executed, the following dialog box appears:*



3. Enter a program name in the Command entry box.
4. Select the Debug check box to run the program under the debugger.
5. Click OK.

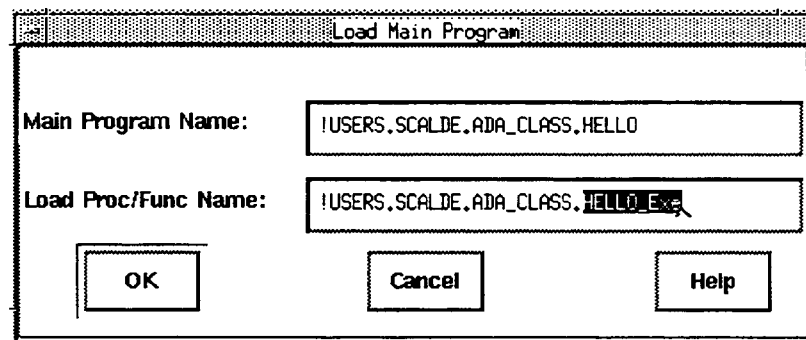
CREATING A LOADED MAIN PROGRAM

A *loaded main program* is an executable program that does not depend on its source code; that is, a loaded main program does not become obsolete if the source code from which it was created is modified. Loaded main programs are useful for frequently used tools and programs. They are also useful for transporting programs between R1000s because they can be moved without having to move and recompile the associated source code.

Loaded main programs are created from *coded main programs*. (A coded main program is a subprogram containing pragma Main that has been promoted to the coded state.) To create a loaded main program from a coded main program:

1. Place the Environment cursor in the coded main program.
2. Choose Program:Build:Load.

The following dialog box appears:



3. Verify the name of the coded main program (from which the loaded main program is to be created) in the Main Program Name entry box.
The specified unit must contain pragma Main and be in the coded state.
4. Enter the name of the loaded main program to be created at the prompt in the Load Proc/Func Name entry box.
5. Click OK.

The load procedure automatically inserts pragma Loaded_Main in place of pragma Main in the newly created Ada specification.

***Note:** Because its code segments are independent of its source code, a loaded main program is unaffected by demoting, and even changing, the source code.*

8

Debugging

This chapter discusses basic debugging operations using the Debugger Palette. For more detailed information about debugging, see the Debugging (DEB) book of the *Rational Environment Reference Manual*.

USING THE DEBUGGER PALETTE

The Debugger Palette allows you to debug Ada programs using a persistent control palette.

The top-left section of the Debugger Palette contains controls that operate the execution of the program running under the debugger. The bottom-left section of the palette contains controls that deal with variables. The right side of the palette contains the debugging option menu that controls sets of buttons for:

- Breakpoints
- Exception Handling
- Task Information
- Task Control
- Debugger Control
- Tracing
- Target
- Machine and Memory
- Quit


To display a set of buttons from the option menu:

1. Click the option menu label to reveal the options.
2. Click the option you want.

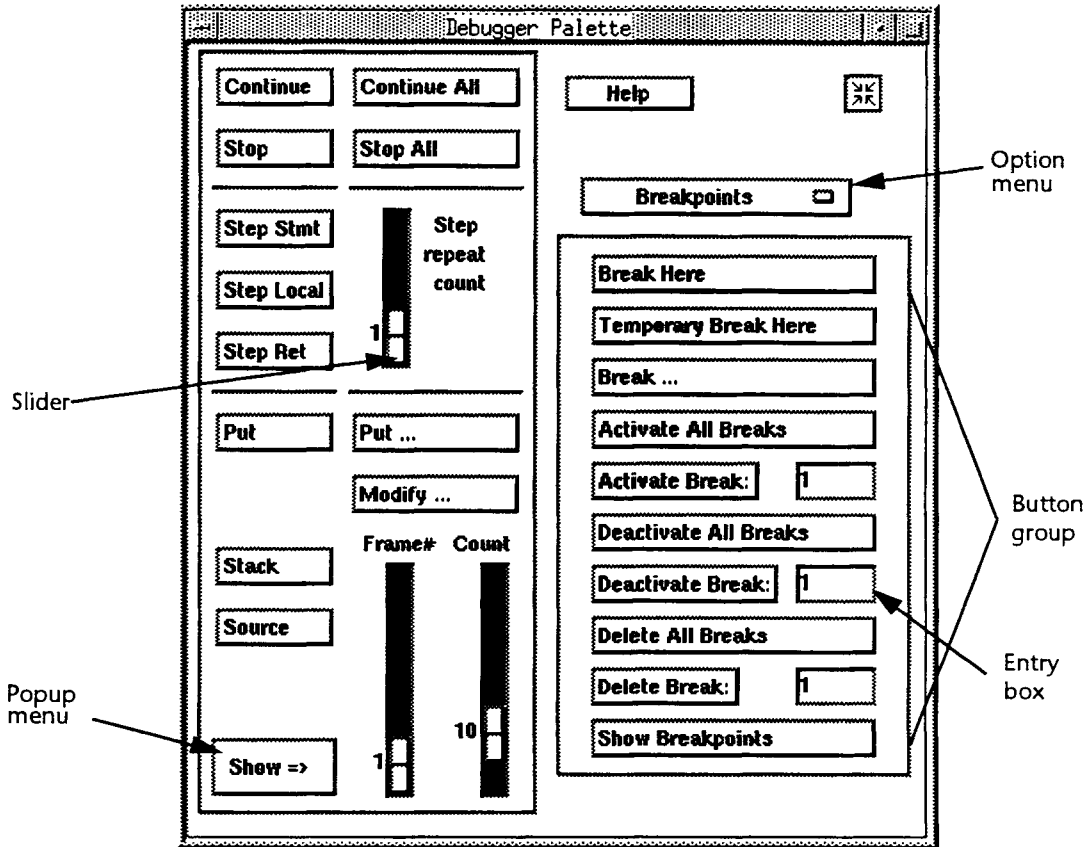
The new set of buttons replaces the old set.

Displaying the Debugger Palette


Use the Debugger Palette to perform all standard debugging operations.

Click: 

Alternative: Choose Debug:Debugger Commands Palette.



Closing the Debugger Palette

On the Debugger Palette, click: 

STARTING THE DEBUGGER

1. Select the Ada unit or part to debug. You may select:
 - A command in a command window
 - A procedure in an Ada unit
 - A main program name
2. Choose Debug:Start Debugging of Command.

The debugger window appears in the Environment area of your Access window.

Alternative: To debug a main program, choose File:Run for that program and click the Debug radio button in the subsequent dialog box. You can also press [Meta][Return] to start the debugger if you have selected the unit to debugged in a command window.

REDISPLAYING THE ENVIRONMENT'S DEBUGGER WINDOW

You can recall the debugger window if it becomes replaced by another window.

Choose `Debug:Debugger Window`.

The debugger window appears in the Environment area of your Access window.

DISPLAYING THE PROGRAM BEING DEBUGGED

A window automatically displays a section of the program around the point where execution was suspended. The statement or declaration to be executed next is highlighted (selected).

STOPPING THE DEBUGGER

Finishing and Killing the Debugging Job

This operation kills the job being debugged and/or the debugger for the session.

Choose `Debug:Finish Debugging Job and Kill`.

Note: The debugger should not need to be killed in normal use.

Finishing and Detaching from the Debugging Job

Choose `Debug:Finish Debugging Job and Detach`.

STEPPING THROUGH THE PROGRAM

You can run the program one step at a time to examine its behavior in detail.

Stepping by a Specific Number of Steps

1. Slide the Step repeat count slider on the Debugger Palette to reflect the number of steps you want to go to.
2. Click the Step Stmt button on the Debugger Palette.

Stepping by Every Statement

Click the Step Stmt button on the Debugger Palette.

A single declaration is elaborated or a single statement is executed. The next statement or declaration to be stepped through is automatically highlighted in the Ada unit.

Stepping without Stopping in Called Subprograms

This operation elaborates declarations and executes statements within each called subprogram without stopping after each individual step.

Click the **Step Local** button on the Debugger Palette. (The cursor can be anywhere on the screen.)

The program is executed or elaborated up to the next statement or declaration at the same level of program structure.

Stepping to the Enclosing Subprogram

Click the **Step Ret** button on the Debugger Palette.

The debugger steps back one level up the stack.

STOPPING A TASK

Stopping the Current Task

Click the **Stop** button on the Debugger Palette.

The task is stopped at the beginning of the next executing statement in that task.

Stopping All Tasks

Click the **Stop All** button on the Debugger Palette.

All tasks are stopped at the beginning of the next executing statement in each task.

CONTINUING A TASK

Continuing the Current Task

Click the **Continue** button on the Debugger Palette.

The debugger will continue with the current task.

Continuing All Tasks

Click the **Continue All** button on the Debugger Palette.

The debugger will continue all tasks.

USING BREAKPOINTS

Setting Breakpoints

1. Display the Ada unit that contains the statement or declaration at which execution is to stop. If the unit has not been displayed already by the debugger, left double-click the mouse or use other traversal operations.
2. Find the last statement or declaration you want executed or elaborated before stopping. Use searching, scrolling, or stepping operations to find the desired program location.
3. Select the next statement or declaration. (If that statement or declaration is the highlighted current location, it is already selected.)
4. With the cursor in the selection, choose **Breakpoints** in the option menu on the Debugger Palette.
5. Click the **Break Here** button.

A breakpoint is set just before the selected location. This means that execution will continue up to but not including the selected location.

***Alternative:** If you know the specific location at which you want to enter a breakpoint, you can choose **Break on the Breakpoints** option menu in the Debugger Palette. Note that this uses debugger naming of the form 1S_2S.*

Showing Breakpoints

1. Choose **Breakpoints** in the option menu on the Debugger Palette.
2. Click the **Show Breakpoints** button.

A list of all breakpoints, active or not, is displayed in the debugger window.

Removing Breakpoints

Removing a Specific Breakpoint

1. Choose **Breakpoints** in the option menu on the Debugger Palette.
2. Enter the number of the breakpoint you want to delete in the **Delete Break** entry box.
3. Click the **Delete Break** button.

The breakpoint is deleted.

Removing All Breakpoints

1. Choose **Breakpoints** in the option menu on the Debugger Palette.
2. Click the **Delete All Breaks** button.

All breakpoints are deleted.

DISPLAYING THE VALUE OF A PROGRAM VARIABLE

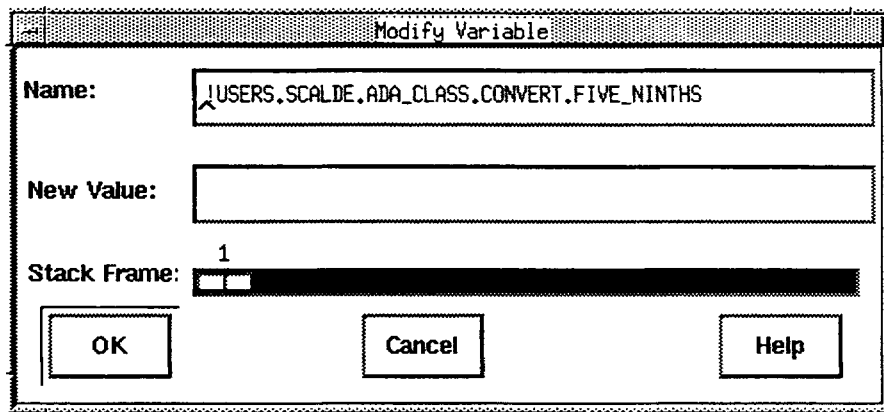
1. Display an Ada unit containing an occurrence of the variable whose value you want to display.
2. Select the occurrence of the variable.
3. Click the left Put button (the one without the ellipsis) on the Debugger Palette.

The selected object and its value are displayed in the debugger window. Formatting is based on the type of object.

MODIFYING VARIABLE VALUES

1. Place the Environment cursor on the variable.
2. Click the Modify button on the Debugger Palette.

The following dialog box appears:



3. Enter the new variable value in the New Value entry box.
4. Click OK to modify the variable.

EXAMINING THE CALL STACK

A *stack frame* contains the values of local variables and parameters to a subprogram. When a subprogram is called, a stack frame is pushed on the stack of the task executing a call.

Displaying the Call Stack

The call stack is the stack of subprogram calls that records the program's flow of control.

Click the Stack button on the Debugger Palette.

The call stack is displayed in the debugger window with the most current call on the top of the stack (it is frame number 1: `_1`).

Displaying Source for a Call-Stack Frame

1. Click the **Stack** button on the Debugger Palette to display the call stack.
2. Place the cursor on the frame you want to display.
3. Click the **Source** button on the Debugger Palette.

The Environment cursor traverses to the specified location in the code of the Ada unit.

Displaying Parameters for a Call-Stack Frame

Displaying the Parameters for the Current Selected Object

By default, this procedure displays the value of the selected object in the frame you designate.

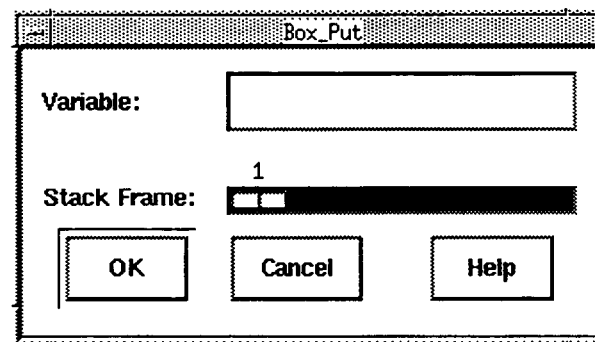
1. Move the **Frame#** slider on the Debugger Palette to the number of the frame you want to search.
2. Click the left **Put** button (the one without the ellipsis) on the Debugger Palette.

The debugger window displays the values that were passed to the selected subprogram at the time it was called.

Displaying Parameters for a Specific Object

1. Click the right **Put** button (the one with the ellipsis) on the Debugger Palette.

The following dialog box appears:



2. Enter the name of the variable whose parameters you want to see in the **Variable** entry box.
3. Move the **Stack Frame** slider to the number of the stack frame you want to search.
4. Click **OK**.

Traversing from the Call Stack

You can view any subprogram referenced in the call stack.

1. Place the cursor in the frame that contains the subprogram you want to view.

2. Double-click the left mouse button.

SETTING UP EXCEPTION HANDLING

Catching Exceptions

Catching Unlisted Exceptions

1. Choose Exception Handling in the option menu on the Debugger Palette.
2. Click the Catch Unlisted button.

Catching Any Exception

1. Choose Exceptions in the option menu on the Debugger Palette.
2. Click the Catch button.

The following dialog box appears:

The screenshot shows a dialog box titled "Debugger Exceptions". It features three text input fields: "Name:", "Task:", and "Location:". Below these fields are three radio buttons: "Catch" (which is selected), "Propagate", and "Forget". At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

3. Enter the name of the exception in the Name entry box.
4. Enter the name of the task that should be monitored for the exception in the Task entry box.
5. To designate a specific location restriction for the exception catch request, enter the name of the location in the Location entry box. By default, exceptions will be caught anywhere they are raised.
6. Click OK.

The debugger stops at the first occurrence of the exception.

Showing Exceptions

1. Choose **Exception Handling** on the option menu.
2. Click the **Show Exceptions** button.

A list of exceptions appears in the debugger window.

RETURNING TO THE POINT OF PROGRAM SUSPENSION

Click the **Source** button.

A window containing the definition of the program being debugged appears. The statement or declaration to be executed next is highlighted.

SHOWING INFORMATION

Showing All Debugger Activities

Choose **All Debugger State** in the **Show** popup menu on the **Debugger Palette**.

A list of debugger activities is displayed in the debugger window.

Showing Libraries

Choose **Libraries** in the **Show** popup menu in the **Debugger Palette**.

A list of libraries in use appears in the debugger window.

Showing Task information

Showing All Tasks

Choose **All Tasks** in the **Show** popup menu in the **Debugger Palette**.

A list of tasks appears in the debugger window.

Showing Stopped Tasks

Choose **Stopped Tasks** in the **Show** popup menu in the **Debugger Palette**.

A list of stopped tasks appears in the debugger window.

Showing Held Tasks

Choose **Held Tasks** in the **Show** popup menu in the **Debugger Palette**.

A list of held tasks appears in the debugger window.

9

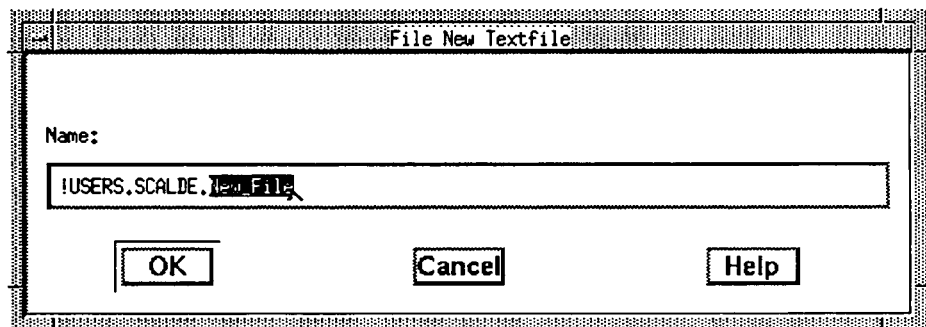
Creating and Modifying Text Files

This chapter describes how to create, modify, and save text files.

CREATING A FILE

1. Place the Environment cursor in the library in which you want the file.
2. Choose **File:New:Text File**.

The following dialog box appears:



3. Enter the name of the new file in the entry box.
4. Click OK.

A new window is created for the image of your file, and an entry for the file appears in the library.

Alternative: Press [Create Text] to create a new text file.

VIEWING A FILE

Viewing a File in the Current Library

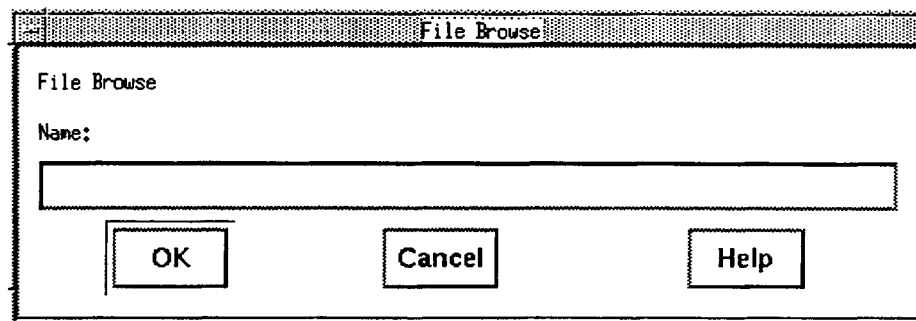
1. Move the mouse pointer to the line containing the file declaration.
2. Double-click the left mouse button.

A window with a read-only image of the file appears.

Viewing a File Located Anywhere

1. Choose File:Browse.

The following dialog box appears:



2. Enter the name of the file you want to view in the entry box.
3. Click OK.

A window with a read-only image of the file appears.

OPENING AN EXISTING FILE FOR EDITING

Place the Environment cursor in the window of the file to be edited.

Choose File:Open.

Note: For specific editing operations, see Chapter 10, "Editing Text."

SAVING A FILE

Closing a File

Place the Environment cursor in the file to be saved.

1. Choose File:Save.
The file is saved.
2. Choose File:Close.

The file is closed for editing and disappears from the Environment area of the Access window.

Saving a File without Closing

Click File:Save.

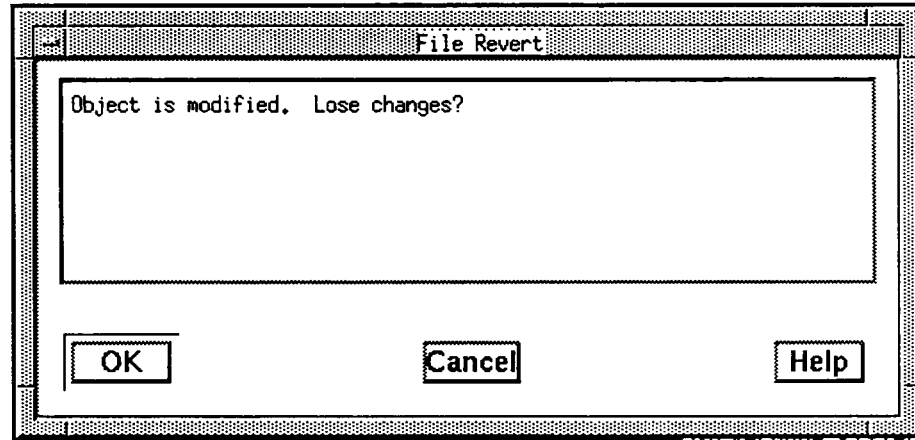
The file remains visible and open for editing.

REVERTING TO THE PREVIOUS VERSION

This command reverts the file to the last saved version.

1. Choose Edit:Revert.

The following dialog box appears:



2. Click OK to revert to the previous saved version.

Note: If this version is saved, revert does not return the previously saved version.

SETTING TABS

1. Press [Cmd Window] to create a command window.
2. To set tab stops at every n th column, enter `set .tab_width(n)`.
3. Press [Promote].

As you edit the text file, pressing [Tab] indents n spaces.

Note: Setting tabs in a particular file affects only that file.

Checking Tabs

1. Place the Environment cursor in the window whose tabs you want to see.
2. Press [Control][Meta][Tab].

Tab markers appear in the message window.

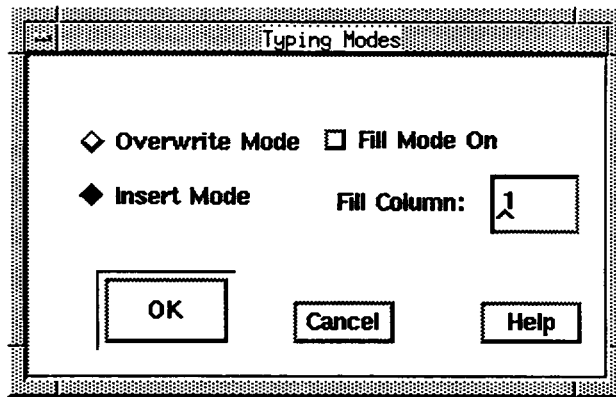
SETTING TYPING MODES

Setting Overwrite Mode

The overwrite mode allows you to type text over existing text.

1. Choose Edit:Typing Modes.

The following dialog box appears:



2. Click the **Overwrite Mode** radio button.

Setting Insert Mode

The insert mode allows you to insert text when you type.

1. Choose **Edit:Typing Modes**.
A dialog box appears (see above).
2. Click the **Insert Mode** radio button.

Setting Wordwrap (Fill Mode)

The fill mode automatically moves the cursor to the next line when it hits a certain column (the default is column number 72).

1. Choose **Edit:Typing Modes**.
2. Click the **Fill Mode** check box.

Fill mode is on when the check box is filled in.

Changing the Wordwrap Column

You can change the column number that marks the spot where the cursor moves to the next line when the fill mode is on.

1. Choose **Edit:Typing Modes**.
A dialog box appears (see above).
2. Enter the new fill column number in the **Set Fill Column** entry box (default is 72).

10

Editing Text

This chapter describes basic Environment text-editing operations. Note that Access allows for both Environment and Motif text selections and operations. Environment text selections are marked in bold, and Motif text selections are represented by reverse video.

MOVING THE ENVIRONMENT CURSOR WITH THE KEYBOARD

Table 10-1 lists the keys that move the Environment cursor within an Environment window.

Table 10-1 *Moving the Environment Cursor with the Keyboard*

Key	Action
[↑] or [Control][U]	Moves cursor up one space
[↓] or [Control][N]	Moves cursor down one space
[←], [Control][H], or [Control][B]	Moves cursor left one space
[→], [Control][J], or [Control][F]	Moves cursor right one space
[Home] or [Control][A]	Moves cursor to the beginning of the line
[End] or [Control][E]	Moves cursor to the end of the line
[Meta][J]	Moves cursor to the next word
[Control][[Moves cursor to the beginning of a region of selected text
[Control][)]	Moves cursor to the end of a region of selected text
[Control][Meta][Home]	Moves cursor to the beginning of the window frame
[Control][Meta][End]	Moves cursor to the end of the window frame

SELECTING TEXT

Selecting Text in the Environment

Selecting a Word

Place the mouse pointer on any character of the word to be selected and [Control]+ double-click the left mouse button.

The word is selected.

Selecting the Preceding Word

Press [Control][↑].

The word before the word that contains the Environment cursor is selected.

Selecting the Next Word

Press [Control][↓].

The word after the word that contains the Environment cursor is selected.

Selecting a Sentence

1. Place the mouse pointer on any word in the sentence.
2. Press [Control] and double-click the left mouse button.
3. Repeat step 2.

The sentence is selected.

Selecting a Paragraph

1. Place the mouse pointer on any part of the paragraph.
2. Press [Control] and double-click the left mouse button.
The word that the cursor is on is selected.
3. Repeat step 2 twice.

First the sentence is selected, and then the entire paragraph is selected.

Selecting an Arbitrary Region of Text

1. Place the pointer at the beginning of the text to be selected.
2. Press [Control] and click the left mouse button.
The Environment cursor marks the beginning of the selected text.
3. Move the pointer to the end of the text to be selected.
4. Press [Control] and click the right mouse button.

The cursor appears at the end of the selected text, which is now bold.

Alternative: Press [Control] and drag the mouse pointer over the text. When you release the mouse button, the text that the pointer passed over is selected. Or place the

pointer at the beginning of the region and press [Control][[]]. Then place the pointer at the end of the region and press [Control][[]].

Selecting All in a File

1. Place the cursor anywhere in the file.
2. Press [Control] and double-click the left mouse button.
The word or white space that the cursor is on is selected.
3. Repeat step 2 four times.

First the sentence the Environment cursor is on is selected, then the paragraph, then any white space between paragraphs, and then the entire file.

Selecting the Parent or Child

Selecting the parent selects successively larger items, and selecting the child selects successively smaller items, based on the following hierarchy:

- Word that the Environment cursor is on
- Sentence
- Paragraph
- Entire file

To select the parent, press [Control][←].

To select the child, press [Control][→].

Note: These operations work differently in an Ada unit. See "Selecting Parent/Child Items in an Ada Unit" in Chapter 7, "Writing Ada Programs."

Deselecting Text

1. Place the Environment cursor anywhere in the selected region.
2. Choose Edit:Deselect.

Any selected text becomes deselected.

Alternative: Press [Control]+click, [Control][X], or [Control][\] to deselect text.

Selecting Text with Motif

Motif selection allows you to select text to be copied to a non-Access window.

Selecting a Region of Text

1. Place the mouse pointer at the beginning of the text you want to select.
2. Drag the pointer over the text.

The selected text is reverse video.

Alternative: Place the Environment cursor at the beginning of the region you want to copy. Then move the pointer to the end of the region and press [Shift] and click the left mouse button.

Deselecting a Region of Text

Click the mouse or move the Environment cursor using the arrow keys.

The region is no longer highlighted. Note that the Motif selection is still retained on the memory buffer until another selection is made.

COPYING SELECTED TEXT

Copying an Environment Selection

1. Choose Edit:Copy.

The text is copied onto the Environment memory buffer.

2. Place the Environment cursor at the exact insertion point where you want to copy the text.
3. Choose Edit:Paste.

The text is inserted at the location of the Environment cursor.

The text can be copied multiple times until you copy another selection onto the memory buffer using the Edit:Copy or Edit:Cut operation.

Alternative: Press [Control][C] to copy the selected text onto the Environment memory buffer, and press [Control][Y] to paste the text.

Copying a Motif Selection

The Motif copy operation allows you to copy text to another X-window or another Access window, as well as within the current Access window.

1. Select the text using the Motif selection operation (see "Selecting Text with Motif," above).
2. Place the Environment cursor at the insertion point where you want to copy text.
3. Click the middle mouse button.

The text is copied at the location of the Environment cursor.

Copying a Line of Text

1. Place the Environment cursor anywhere on the line.
2. Press [Control][Meta][C].

The line is copied directly below. The cursor remains in its original position.

MOVING SELECTED TEXT

1. Choose Edit:Cut.

The selected text is deleted from the screen.

2. Place the Environment cursor at the exact insertion point where you want to move the text.
3. Choose Edit:Paste.

The text is inserted at the location of the Environment cursor.

The text can be copied multiple times until you copy another selection onto the memory buffer using the Edit:Copy or Edit:Cut operation.

SEARCHING FOR AND REPLACING TEXT

Searching for a String

1. Choose Edit:Search/Replace.

The following dialog box appears:

The dialog box titled "Search and Replace" contains the following elements:

- Search For:** A text input field.
- Change To:** A text input field.
- Search Options:**
 - Consider Case
 - Preserve Case
 - Wildcards
 - Current Selection
- Search Direction:**
 - Forward
 - Backward
- Buttons:** Search, Replace, Replace & Search, Replace All, Close, Help.

2. Enter the string to be searched for in the Search For entry box.
3. Click the Forward or Backward radio button to specify the direction of the search.
4. Press the Search command button.

The cursor moves to the first occurrence of the string.

5. Press the Search command button to go to the next occurrence.

Note: No wraparound search occurs.

Alternative: Press [Control][S] to begin searching forward from the Environment cursor. Next, enter the string at the Search: prompt that appears in the message window. Press [Control][S] again to search for the first occurrence and repeat to search for other

occurrences. To search backward from the Environment cursor, press [Control][R] instead.

Searching and Replacing a String

1. Choose Edit:Search/Replace.
A dialog box appears (see above).
2. Enter the string to be replaced in the Search For entry box.
3. Enter the new string in the Change To entry box.
4. Click the Forward or Backward radio button to specify the direction of the search.
5. Click the Replace & Search command button in the dialog box.
The new entry replaces the first occurrence of the string and moves to the next occurrence.
6. Continue to click the Replace & Search command button to replace the next occurrences of the string.

Alternative: Press [Control][Meta][S] to begin the search and replace operation. Next, enter the string to search for at the Search: prompt in the message window and the replacement string at the Replace: prompt. Press [Control][Meta][S] to replace the first occurrence and repeat for other occurrences. To search and replace backward from the Environment cursor, press [Control][Meta][R].

Searching and Replacing All Occurrences of a String

1. Choose Edit:Search/Replace.
A dialog box appears (see above).
2. Enter the string to be replaced in the Search For entry box.
3. Enter the new string in the Replace & Search entry box.
4. Click the Forward or Backward radio button to specify the direction of the search.
5. Press the Replace All command button in the dialog box.

The new string replaces all occurrences of the original string.

Search Options

- **Consider Case:** Replaces text with the case of the text entered in the Change To field.
- **Preserve Case:** Keeps the case of the text as it finds it in the string that is being replaced.
- **Wildcards:** Treats wildcard characters as wildcards. When this option is not selected, all characters are treated as literal characters.
- **Current Selection:** Searches only selected text.

Note: If you do not have a session-switch file for your current session, you cannot make the search case-sensitive. See the *Session and Job Management (SJM)* book of the Rational Environment Reference Manual for more information on session-switch files.

DELETING TEXT

Table 10-2 lists various ways to delete text.

Table 10-2 *Deleting Text*

Text to Be Deleted	Operation
Selected region of text	Edit:Cut, [Control][W], [Shift][Delete], or [Shift][Backspace]
Character the Environment cursor is on	[Control][D]
Previous character	[Delete] or [Backspace]
Entire word the Environment cursor is on	[Meta][D]
From the Environment cursor to the end of the word	[Meta][K]
Entire line	[Control][Meta][D]
From the Environment cursor to the end of the line	[Control][Delete] or [Control][Backspace] or [Control][K]
White space	[Control][Meta][Backspace] or [Control][Meta][Delete]

TRANSPOSING TEXT

Transposing Characters

This operation switches the character that the cursor is on with the character preceding.

1. Place the cursor on the second character.
2. Press [Control][T].

The characters transpose. The Environment cursor remains in its original position.

Transposing Words

This operation switches the word that the cursor is on with the preceding word. Word terminators are blanks, underscores, semicolons, or periods.

1. Place the cursor anywhere on the second word.
2. Press [Meta][T].

The words transpose. The Environment cursor remains in its original position.

Transposing Lines

This operation switches the line that the cursor is on with the preceding line.

1. Move the cursor to any place on the lower line.
2. Press [Control][Meta][T].

The lines transpose. The Environment cursor remains in its original position.

CHANGING THE CASE OF TEXT

Making Text Uppercase

Making a Word Uppercase

This operation affects the text from the location of the cursor to the end of the word.

1. Place the Environment cursor on the first letter of the word, or the space directly before, to make the entire word uppercase.
2. Press [Meta][.] or [Meta][>].

The word becomes uppercase, and the cursor traverses to the first space after the word.

Making a Selected Region of Text Uppercase

1. Select the text that you want to modify.
2. Choose Edit:Uppercase.

All selected text becomes uppercase.

Making Text Lowercase

Making a Word Lowercase

This operation affects the text from the location of the cursor to the end of the word.

1. Place the Environment cursor on the first letter of the word, or the space before, to make the entire word lowercase.
2. Press [Meta][,] or [Meta][<].

The word becomes lowercase, and the cursor traverses to the first space after the word.

Making a Selected Region of Text Uppercase

1. Select the text that you want to modify.
2. Choose Edit:Lowercase.

All selected text becomes lowercase.

Making Text Capitalized

Making a Word Capitalized

1. Place the Environment cursor on the first letter of the word, or the space directly before.
2. Press **[Meta][^]** or **[Meta][6]**.

The first letter of the word is capitalized.

Making a Selected Region of Text Capitalized

1. Select the text that you want to capitalize.
2. Choose **Edit:Capitalize**.

The first letter of every word of selected text is capitalized, and the cursor traverses to the first space after the region.

FILLING A REGION OF TEXT

This operation fills in the white space in a selected region of text, leaving two spaces between sentences.

1. Select a region of text.
2. Choose **Edit:Fill**.

The region is filled.

JUSTIFYING A REGION OF TEXT

1. Select the text to be justified.
2. Choose **Edit:Justify**.

The text becomes right-justified. The column number default is 72.

GETTING LINE INFORMATION

Press **[Control][Home]**.

The message window displays the line and column number of the Environment cursor and the total number of lines in the file.

SAVING CHANGES

Saving Changes One Image at a Time

1. Begin with the cursor in the window of the image to be saved.
2. Choose **File:Save**.

The object is saved and its image remains open for editing.

Alternative: Press [Shift][Enter] or [Control][Return] to save changes.

Saving Changes in All Images in a Single Operation

1. Press [Cmd Window] to create a command window.
2. Enter `Window.Directory` and press [Promote].

The window directory appears in an Environment window and lists all Environment images that you have opened during your session. Images with unsaved changes have a * in the Mod column.

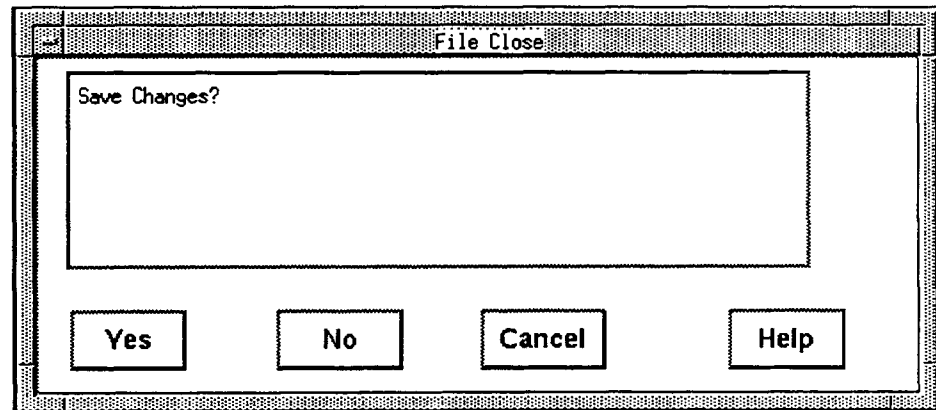
3. Place the Environment cursor on the top line of the image, designating all the objects listed in the window directory.
4. Choose **File:Save**.

All objects that have been changed are saved and now have a blank in the Mod column of the window directory.

Saving the Image and Closing the File

1. Choose **File:Close**.

The following dialog box appears:



2. Click **Yes** to save the changes.

11

Managing Libraries

Libraries are the basic structuring mechanism in the Environment. They can be used to separate different projects, parts of projects, test cases, documentation, and so on. In this sense, they are similar to directories on other systems. The library system in the Environment is a nested hierarchy of several kinds of libraries:

- *Directories* are the most basic kind of library. For the most part, directories assume the characteristics of their parent world or view. For this reason, they are generally used only to partition groups of objects.
- *Worlds* are the control point for certain Environment resources, including resources for managing disk volume, access rights, and Ada compilation. Worlds determine the units that are potentially visible to the Ada units in them and in nested subdirectories and determine the target for which units will be compiled.
- *Subsystems* and *subsystem views* are special-purpose libraries for use with the Environment's facilities for configuration management and version control (CMVC). For the most part, views have the same features as worlds. (For more information about subsystems and views, see Chapter 12, "Using CMVC.")

Any kind of library can contain any or all of the following objects:

- Other libraries. For the most part, the nesting of libraries is arbitrary; however, subsystems and subsystem views can contain only directories.
- Ada compilation units.
- Files, including activity, binary, switch, text, and others.
- Other objects such as sessions.

This chapter describes how to create and maintain libraries and library objects in the Environment. For more information about the Environment library system, see the Library Management (LM) book of the *Rational Environment Reference Manual*.

CONTROLLING THE LIBRARY DISPLAY

Toggling Information on Library Objects

1. Place the Environment cursor on the top line of the library.
2. Press [Explain].

Repeating this command toggles the library display so that you view one of the following:

- The default level of detail, containing only the name of each object

- The standard level of detail, containing the name, class, subclass, and unit state of each object
- The miscellaneous level of detail, containing the name, unit state, date and time of last update, user who last updated, size in bytes, number of deleted versions to be retained, and an indication if the object is frozen for each object

Note that you can change the amount of detail you see when you first log in and the specific information displayed in the standard and miscellaneous levels of detail by editing your session switches (see Session Switches in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual*).

Showing More Objects in the Library

At any level of detail (default, standard, miscellaneous), a library display can also be adjusted “vertically” (*expanded* and *elided*) to show more or fewer objects in the library. At the system-defined default level of elision, the default versions of all undeleted objects are displayed. To display additional objects:

- Press [Meta][!]

Repeatedly pressing [Meta][!] expands the display through several levels. These levels are indicated in the window banner:

- {versions} indicates all deleted and undeleted versions.
- versions indicates all undeleted versions.
- {lib vers} indicates all deleted and undeleted versions; no subunits.
- lib vers indicates all undeleted versions; no subunits.
- {units} indicates all deleted and undeleted objects.
- units indicates all undeleted objects.
- {lib units} indicates all deleted and undeleted objects; no subunits.
- lib units indicates all undeleted objects; no subunits.

Showing Fewer Objects in the Library

At any level of detail (default, standard, miscellaneous), a library display can also be adjusted “vertically” (*expanded* and *elided*) to show more or fewer object in the library. At the system-defined default level of elision, the default versions of all undeleted objects are displayed. To display fewer objects:

- Press [Meta][)]

Repeatedly pressing [Meta][)] expands the display through several levels. These levels are indicated in the window banner and described above, in “Showing More Objects in the Library.”

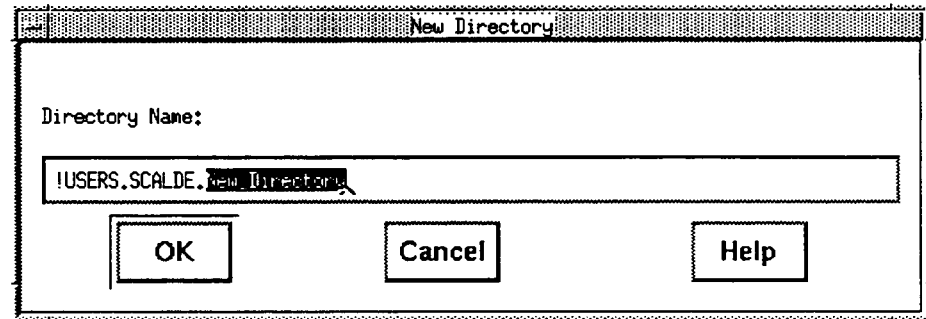
CREATING NONSUBSYSTEM LIBRARIES

To create subsystems and subsystem views, which are also libraries, see Chapter 12, “Using CMVC.”

Creating a Directory

1. Place the Environment cursor in the library that is to contain the new directory.
2. Choose File:New:Directory.

The following dialog box appears:



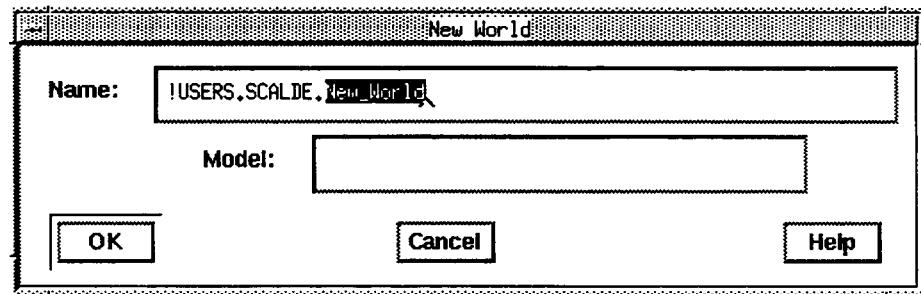
3. Enter the name of the new directory in the Name entry box.
4. Click OK.

The Environment creates the directory. In the enclosing library, you see the new directory name inserted in alphabetical order.

Creating a World

1. Place the Environment cursor in the library that is to create the new world.
2. Choose File:New:World.

The following dialog box appears:



3. Enter the name of the new world in the Name entry box.
4. To create a world based after a specific model, enter the name of the model world in the Model entry box. When creating a world, you can specify a *model world* for the new world to copy. In particular, model worlds determine the initial set of links, the library-switch file, and the target key associated with the new world.
5. Click OK.

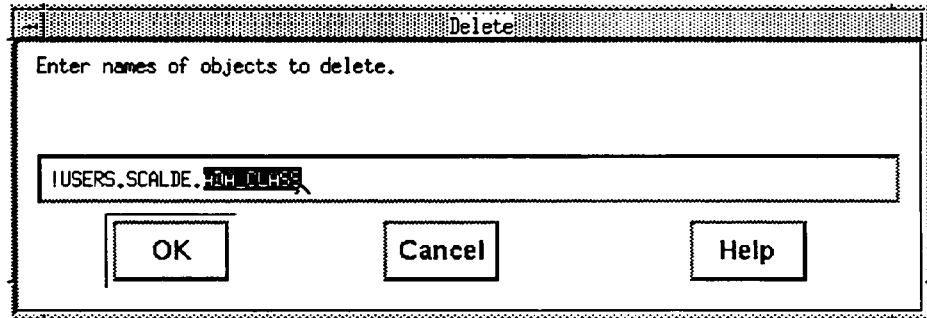
The Environment creates the world. In the enclosing library, you see the name of the new world inserted in alphabetical order.

DESTROYING OBJECTS

Caution: This operation is unrecoverable. Destroyed objects cannot be recovered, regardless of the retention count of the object or enclosing library. To remove objects while maintaining a recoverable version, see the *Library.Delete* procedure in the *Library Management (LM)* book of the Rational Environment Reference Manual.

1. Select the name of the object (or one of multiple objects) to be destroyed.
2. Choose File:Delete File.

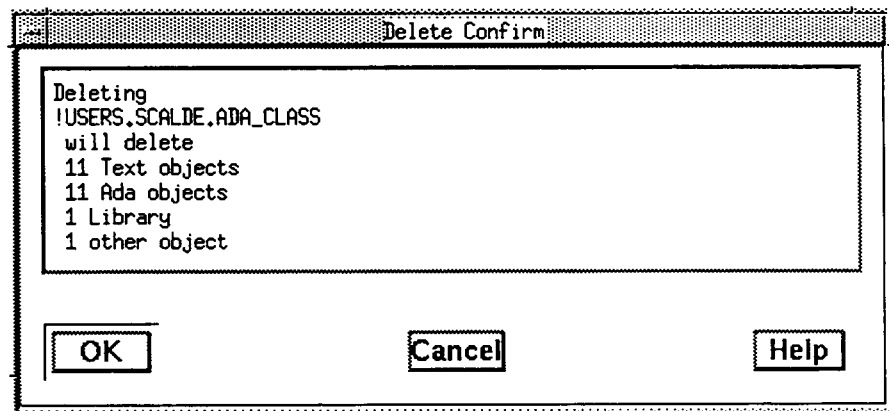
The following dialog box appears:



3. Verify the name of the object(s) to be destroyed:
 - To destroy a single object, specify the name of that object.
 - To destroy multiple objects, specify an appropriate naming expression using set notation ([object1, object2]) or the standard wildcard characters for specifying pathnames:
 - # matches any single character.
 - @ matches zero or more characters.
 - ? matches zero or more nonworld name components.
 - ?? matches zero or more name components, including worlds.
4. Click OK.

If you have specified a single object to be destroyed and no other objects would be affected (such as dependent Ada units), that object is destroyed.

If you have specified multiple objects or there are other objects that would be affected by the deletion, a second dialog box appears with a list of the number of libraries, text objects, and other objects that will be destroyed:



- Verify that the appropriate number and kind of objects are listed.

If you named a library to be destroyed, this operation destroys the library and all of its contents, including other libraries, files, and Ada units.

If you named an Ada unit to be destroyed, this operation destroys:

- The named unit(s)
- The corresponding subunits when unit bodies are named
- The corresponding body and subunits when unit specifications are named

If you think the wrong set of objects will be destroyed, click Cancel and try the operation with a different name or naming expression in the entry box.

If the correct set of objects is listed, click OK.

The named objects and all of their child objects are destroyed. Destroying an object permanently deletes and expunges all versions of the specified object, regardless of the retention count for the object or the enclosing library. A destroyed object has no entry in the library display and cannot be recovered.

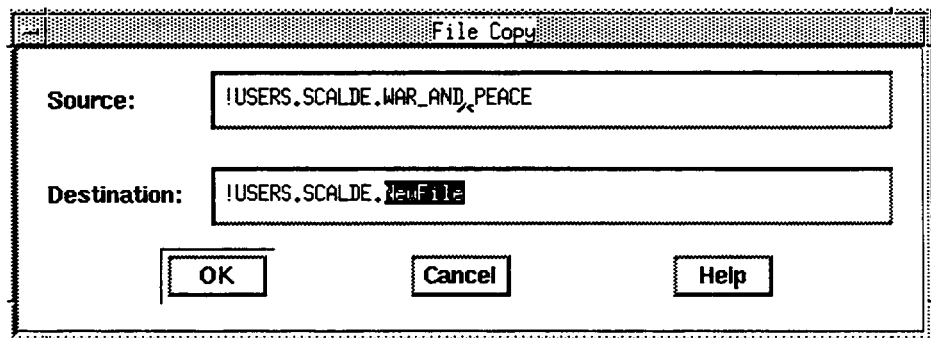
Any dependent Ada units are demoted. In particular, this operation:

- Demotes any units that *with* the deleted units
- Demotes any units that *with* the demoted units

COPYING OBJECTS

- Select the name of the object (or one of multiple objects) to be copied.
- Traverse to the destination library.
- Choose File:Copy File.

The following dialog box appears:



- Verify the name of the Source object(s) to be copied:
 - To copy a single object, specify the name of that object.
 - To copy a library and all of its contents, specify the name of that library.
 - To copy multiple objects, specify an appropriate naming expression using set notation ([object1, object2]) or the standard wildcard characters for specifying pathnames:
 - # matches any single character.
 - @ matches zero or more characters.
 - ? matches zero or more nonworld name components.
 - ?? matches zero or more name components, including worlds.

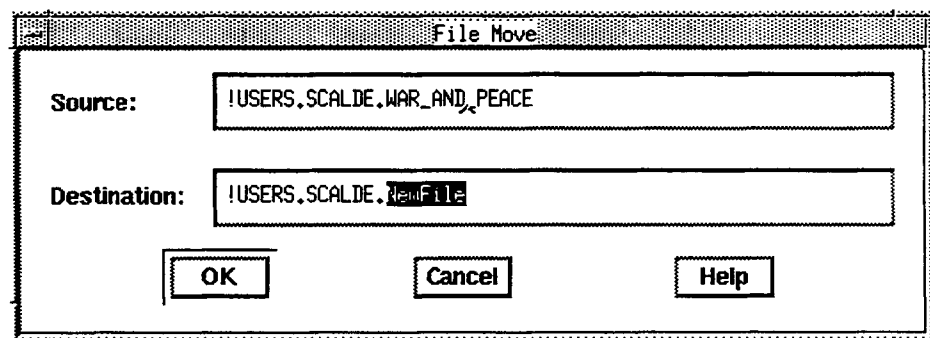
5. Verify the **Destination** to which the object(s) are to be copied:
 - To copy the objects into a particular library, while preserving their simple names, specify the full name of that library.
 - To copy the objects and change their simple names, specify the full path-name, including the new simple names. You can use the following substitution characters:
 - @ expands to the string(s) matched by a wildcard in the **Source** entry box.
 - # expands to a name component from the **Source** entry box. Name components are matched from right to left.
6. Click **OK**.

See "Parameter-Value Conventions" in the Reference Summary (RS) book of the *Rational Environment Reference Manual* for more information and examples using set notation, wildcard characters, and substitution characters.

MOVING OR RENAMING OBJECTS

1. Select the name of the object (or one of multiple objects) to be moved.
2. Traverse to the destination library.
3. Choose **File:Move File**.

The following dialog box appears:



4. Verify the name of the **Source** object(s) to be moved:
 - To move a single object, specify the name of that object.
 - To move a library and all of its contents, specify the name of that library.
 - To move multiple objects, specify an appropriate naming expression using set notation ([object1, object2]) or the standard wildcard characters for specifying pathnames:
 - # matches any single character.
 - @ matches zero or more characters.
 - ? matches zero or more nonworld name components.
 - ?? matches zero or more name components, including worlds.
5. Verify the **Destination** to which the object(s) are to be moved:
 - To move the objects into a particular library, while preserving their simple names, specify the full name of that library.
 - To move the objects and change their simple names, specify the full path-name, including the new simple names. You can use the following substitution characters:

- @ expands to the string(s) matched by a wildcard in the Source entry box.
 - # expands to a name component from the Source entry box. Name components are matched from right to left.
 - To rename the objects, simply enter the new simple name, while preserving the original library pathname.
6. Click OK.

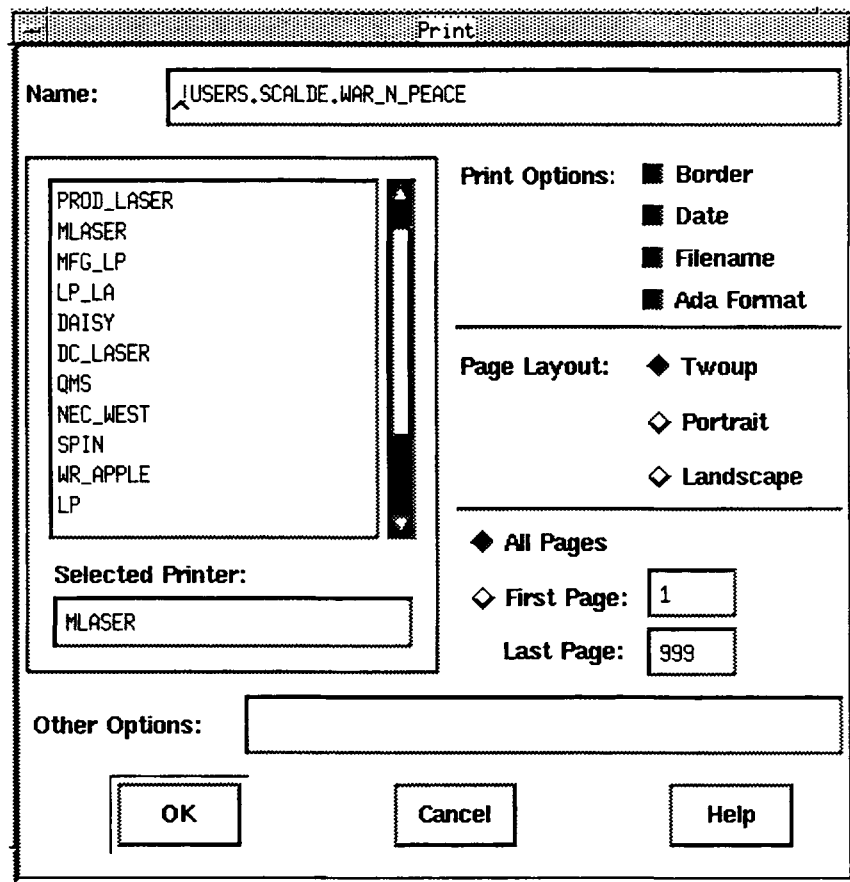
The progress of the command is displayed in an Environment I/O window. Ada units are demoted to source. The old object name is removed from the library and replaced by the new name, which is inserted in the library in alphabetical order.

See "Parameter-Value Conventions" in the Reference Summary (RS) book of the *Rational Environment Reference Manual* for more information and examples using set notation, wildcard characters, and substitution characters.

PRINTING OBJECTS AND IMAGES

1. Select the name of the object to be printed or place the Environment cursor in its image.
2. Choose File:Print.

The following dialog box appears:



3. Click OK to print using the default options.

Printing Multiple Objects

To print multiple objects, enter the names of the objects in the Name entry box of the Print dialog box (see above), using an appropriate naming expression.

Table 11-1 Naming Expressions for Printing Multiple Objects

Characters	Description
#	Matches a single character other than a period: T#### matches Tools.
@	Matches zero or more characters not containing a period: !U@.@.Tools matches !Users.Anderson.Tools.
?	Matches zero or more nonworld name components: !Users.Anderson? matches !Users.Anderson and all the objects in it except worlds.
??	Matches zero or more name components of any kind: !Users?? matches !Users, all home worlds, and all their contents.
[. . .]	Encloses a set of names: [!Users.Anderson.Letter, !Users.Anderson.Notes] matches two files, Letter and Notes.
~name	Excludes a name from a set: [@, ~Tools] matches everything except Tools.

Selecting the Printer

Click the name of the printer you want to use in the printer listing. (Note that this list is built from the printer-configuration information set up through !Machine.Initialization. If your system manager has set up printer configurations using another mechanism, this list will be empty.)

The name appears in the Selected Printer entry box.

Alternative: Directly type in the name of the printer in the Selected Printer entry box.

Specifying the Pages to Print

Printing All Pages in a File

Click the All Pages check box in the Print dialog box.

Printing Specific Pages

1. Click the First Page check box in the Print dialog box.
2. Enter the number of the first page of the file you want to print in the First Page entry box. Note that this operation assumes you have entered the physical page number, not the page number that will be printed on the page.
3. Enter the number of the last page you want to print in the Last Page entry box.
To print one page of a file, enter that page number in both entry boxes.
4. Click OK to print the page(s).

Format Options

- **Border:** Prints a border around the page. If the **Twoup** option (see below) is also selected, a border is printed around both text areas on the page.
- **Date:** Prints the date and time of the printout at the bottom of each page.
- **Filename:** Prints the full pathname of the file at the top of each page.
- **Ada Format:** Pretty-prints Ada reserved words in bold when printing source code.

Page-Layout Options

- **Twoup:** Prints two text areas (in book form) across the length of the page.
- **Portrait:** Prints the page vertically.
- **Landscape:** Prints the page horizontally.

Other Options

You can enter specific print options in the **Other Options** entry box. See package **Queue** in the *Session and Job Management (SJM)* book of the *Rational Environment Reference Manual* for more information.

12

Using CMVC

The Rational Environment provides support for project management through its system of configuration management and version control (CMVC). CMVC provides support for:

- Project partitioning: You can break a project into a manageable number of higher-level components called *subsystems*, each containing a group of logically related objects. For Ada programs, subsystems are units of decomposition similar to, but larger than, the Ada package, which preserve on a larger scale the Ada notion of separate specification and implementation.
- Version control: You can control and track changes to individual objects within each subsystem and record what changes are made and why they were made.
- Configuration management: You can construct, release, and maintain multiple consistent sets (or *configurations*) of versions within each subsystem. (Each alternative configuration constitutes a *view* of that subsystem.) At a higher level, configuration management refers to combining views from each subsystem in order to create entire applications.

This chapter describes basic CMVC operations for creating and maintaining subsystems and subsystem views. For more information about creating, manipulating, and maintaining subsystems, see the Key Concepts of the Project Management (PM) book of the *Rational Environment Reference Manual*.

CREATING A SUBSYSTEM

Subsystems are a kind of library used to encapsulate a program's compilation units by grouping Ada units or other objects. Subsystems are more powerful than other libraries for the following reasons:

- Subsystems, like Ada packages, provide a means for defining and enforcing interfaces among an application's components. These interfaces provide explicit control over dependencies among units in different subsystems.
- Subsystem interfaces impose explicit bounds on the recompilation required after changes are made to the implementation.
- Subsystems provide a mechanism for developing alternative implementations of an application's components. Execution and testing of the entire application is a matter of specifying the desired combination of precompiled implementations, one from each subsystem within an application.
- CMVC operations are available within subsystems for tracking unit changes, coordinating access to shared units, and propagating changes across shared units.

To create a subsystem:

1. Choose File:New:Subsystem.

The following dialog box appears:

2. In the first Name entry box, enter the name of the subsystem to be created. The default is *Current_Library.New_Subsystem*.
3. In the Initial View Name entry box, enter the name of the initial working view to be created. The default is *Rev1_Working*.
4. From the Kind option menu, choose the kind of subsystem to create. The two kinds of subsystems determine the kinds of views that can be created as well as whether hierarchic importing is enforced:
 - **Spec_Load** creates a subsystem that can contain either *spec* and *load views* or *combined views*:
 - A load view contains an implementation of a subsystem.
 - A spec view expresses a subsystem's exports.
 - A combined view both contains a subsystem implementation and expresses the exports from that implementation.

Within a spec/load subsystem, all imports must be hierarchic, in that no view is permitted to be in its own import closure.
 - **Combined** creates a subsystem that can contain only *combined views*, among which circular import relations may hold.
5. In the Model entry box, enter the name of the model from which to create the initial working view. The model determines the initial links, the settings in the *Compiler_Switches* file, and the target key associated with the view. The default is *!Model.R1000*.
6. In the Comments entry box, enter any comments, such as purpose of the subsystem.
7. Click OK.

The progress of the procedure is displayed in an I/O window. The newly created subsystem contains three libraries: Configurations (a directory that contains summary information about each view in the subsystem), Rev1_Working (a program library in which your ongoing work takes place), and State (a directory that contains information about the underlying objects in the subsystem).

Making a Path

Separate development efforts within a single subsystem are maintained in multiple development *paths*. When you create a subsystem, the Environment automatically creates one working view that serves as the primary path. You can create additional paths by creating other working views from an existing view.

To create a path:

1. Place the Environment cursor in the subsystem in which you want to make a path.
2. Choose File:New:Working View.

The following dialog box appears:

3. In the Name entry box, enter the simple name of the new path. If the parent path is Rev1_Working, the default name is usually of the form Rev1_1_Working.
4. From the Kind option menu, choose the kind of view to create.
 - **Spec_Load** creates a load view. A load view contains the implementation of a subsystem. Using spec and load views minimizes the recompilation required after changes are made and eliminates the need for recompilation during recombinant testing. (To create a spec view, see “Making a Spec View” on page 91.)

- **Combined** creates a combined view. A combined view both contains the subsystem implementation and expresses the exports from that implementation. Using combined views does not reduce the recompilation requirements; however, you must use combined views when generics or inlined subprograms are exported from implementations for non-R1000 targets.
5. Decide whether or not to join the new path to the parent path and check the **Join** check box, as decided:
 - You should join the two paths if most of the controlled objects in them are to be joined. Joined objects cannot be checked out or modified independently.
 - You should not join the two paths if most of the controlled objects in the two paths are to be worked on independently.
 6. Enter or edit the name of the parent path in the **Copy Of** entry box. The pathname can be:
 - A combined or load view (not a spec view)
 - A working view or a release view
 All units in the parent path must be checked in.
 7. Choose either the **Same Imports** or **New Imports** radio button:
 - If you choose **Same Imports**, the new view will have the same imports as the parent view, regardless of the contents of the **New Imports** entry box.
 - If you choose **New Imports**, the new view will have only the imports you specify in the associated entry box.
 8. Enter any comments in the **Comments** entry box.
 9. In the **Model** entry box, enter the model from which to create the new view. The model determines the initial links, the settings in the **Compiler_Switches** file, and the target key associated with the view. The default is **!Model.R1000**. By default, the model is inherited from the parent view.
 10. Click **OK**.

The command displays messages in an I/O window. When it completes, a new working view, indicating the new path, appears in the subsystem.

Making a Subpath

When a team is assigned to implement a subsystem, a separate *subpath* can be created for each individual on the team. Subpaths are working views in which changes can be made and tested; they are created as full copies from the path's working view.

To create a subpath:

1. Place the **Environment** cursor in the subsystem in which to create the subpath.
2. Choose **File:New:Working View**.
The **New Working View** dialog box appears (see above).
3. Enter the full name of the subpath in the **Name** entry box.
If the path is **Rev1_Working**, the subpath usually has a name in the form of **Rev1_Subpath_Working**.
4. Choose **Spec_Load** or **Combined** in the **Kind** option menu (usually you will want to make the subpath the same kind as the parent path).
5. Fill in the **Join** check box if it is not already selected.

6. Enter or edit the name of the parent path in the **Copy Of** entry box.
All units must be checked in so they can be joined with the new subpath.
7. Choose either the **Same Imports** or **New Imports** radio button.
If you choose **New Imports**, enter the new imports in the associated entry box.
8. Enter any comments in the **Comments** entry box.
9. To base the new subpath on a particular model, enter the model name in the **Model** entry box.
10. Click **OK**.

The command displays messages in an I/O window. When it completes, a new view appears in the subsystem that is the working view for the subpath.

Making a Spec View

In a spec/load subsystem, a spec view defines the set of implemented units that are potentially available, or visible, to units in views of other subsystems. Spec views thus define a subsystem's exports; as such, spec views can be imported by client views in other subsystems. In a sense, a spec view is analogous to an Ada package specification, which defines the resources that are available to client units.

To create a spec view:

1. Place the Environment cursor in the subsystem in which to create the spec view.
2. Choose **File:New:Spec View**.

The following dialog box appears:

The dialog box titled "New Spec View" contains the following elements:

- Name:** A text box containing "REV1_1_Spec".
- Join:** An unchecked checkbox.
- Copy Of:** A text box containing "!USERS.SCALDE.MAIL_UTILITIES.REV1_WORKING".
- Same Imports:** A section header with a diamond icon.
- New Imports:** An empty text box.
- Comments:** An empty text area.
- Model:** A text box containing "!MODEL.R1000".
- Buttons:** "OK", "Cancel", and "Help" buttons at the bottom.

3. In the **Name** entry box, you can edit the name of the spec view.
4. In the **Copy Of** entry box, verify the name of the view to be copied.

5. Select the **Same Imports** or **New Imports** radio button.
If you choose **New Imports**, enter the new imports in the associated entry box.
6. Enter any comments in the **Comments** entry box.
7. In the **Model** entry box, enter the name of the model from which to create the initial working view. The model determines the initial links, the settings in the **Compiler_Switches** file, and the target key associated with the view. The default is `!Model.R1000`.
8. Click **OK**.

RELEASING CONFIGURATIONS

As you develop units in a working view, you can preserve certain significant combinations (*configurations*) of generations. You can do this by making a *release* from the working view. A release typically represents a baseline configuration that has been compiled and tested, and thus is considered stable and usable for execution by other subsystems. Releases also can serve as reference points in the development history of a single subsystem.

Several kinds of releases can be made, depending on your needs:

- *Release views* (also called *full-view releases*), which preserve both the source code and the compilation information to permit execution.
- *Configuration releases*, which preserve enough information about configuration state to permit reconstruction of release views.
- *Code views* (also called *code-only releases*), which permit execution without making program source code available. Code views typically are made from a working view for use by the developers of other subsystems, particularly when the subsystems are developed on different R1000s.

Making a Release View

A release view is a complete frozen copy of a working view. As such, a release view contains program source code, and, if the release view has been compiled, the units in the release view can be executed. You should make a release view from a compiled working view whenever you want to both preserve a configuration in a working view and be able to execute its units. (Note that the original working view itself is not frozen, so it is always available for further development.)

To make a release view:

1. Place the Environment cursor in the subsystem in which to create the release view.
2. Choose **File:New:Release View**.

The following dialog box appears:

3. In the Name entry box, enter the simple name of the release view. If the source view is Rev1_Working, the default name is usually of the form Rev1_1.
4. From the Kind option menu, choose the kind of view to create. Usually you will want to choose the same kind as the source view:
 - Spec_Load creates a load view. A load view contains the implementation of a subsystem.
 - Combined creates a combined view. A combined view both contains the subsystem implementation and expresses the exports from that implementation.
5. Decide whether or not to join the release view to the source view and check the Join check box, as decided:
 - You should join the two views if most of the controlled objects in them are to be joined. Joined objects cannot be checked out or modified independently.
 - You should not join the two paths if most of the controlled objects in the two paths are to be worked on independently.

6. Enter or edit the name of the source view in the **CopyOf** entry box. The pathname can be:
 - A combined or load view (not a spec view)
 - A working view or a release viewAll units in the source view must be checked in.
7. Choose either the **Same Imports** or **New Imports** radio button:
 - If you choose **Same Imports**, the release view will have the same imports as the source view, regardless of the contents of the **New Imports** entry box. Usually you will want the release view to have the same imports as the source view.
 - If you choose **New Imports**, the release view will have only the imports you specify in the associated entry box.
8. Enter any comments in the **Comments** entry box.
9. In the **Model** entry box, enter the model from which to create the release view. The model determines the initial links, the settings in the **Compiler_Switches** file, and the target key associated with the view. The default is **!Model.R1000**. By default, the model is inherited from the source view.
10. Check the **Build Full View Copy** box.
11. Click **OK**.

Making a Configuration Release

A configuration release preserves the state of a working view, without creating a release view. As such, a configuration release is a summary of configuration information from which a release view subsequently can be constructed, if desired. You should make a configuration release when you want to keep a record of a particular configuration, but you do not need to execute the units immediately. Making a configuration release is faster and uses less storage than making a release view.

To make a configuration release:

1. Place the **Environment** cursor in the subsystem in which to create the configuration release.
2. Choose **File:New:Release View**.

The following dialog box appears:

3. In the **Name** entry box, enter the simple name of the configuration release. If the source view is `Rev1_Working`, the default name is usually of the form `Rev1_1`.
4. From the **Kind** option menu, choose the kind of view to create. Usually you will want to choose the same kind as the source view:
 - **Spec_Load** creates a load view. A load view contains the implementation of a subsystem.
 - **Combined** creates a combined view. A combined view both contains the subsystem implementation and expresses the exports from that implementation.
5. Enter or edit the name of the source view in the **Copy Of** entry box. The pathname can be:
 - A combined or load view (not a spec view)
 - A working view or a release view
 All units in the source view must be checked in.
6. Choose either the **Same Imports** or the **New Imports** radio button:
 - If you choose **Same Imports**, the configuration release will have the same imports as the source view, regardless of the contents of the **New Imports** entry box. Usually you will want the configuration release to have the same imports as the source view.
 - If you choose **New Imports**, the configuration release will have only the imports you specify in the associated entry box.
7. Enter any comments in the **Comments** entry box.

8. In the **Model** entry box, enter the model from which to create the configuration release. The model determines the initial links, the settings in the `Compiler_Switches` file, and the target key associated with the view. The default is `!Model.R1000`. By default, the model is inherited from the source view.
9. Check the **Save Source in Database Only** box.
10. Click **OK**.

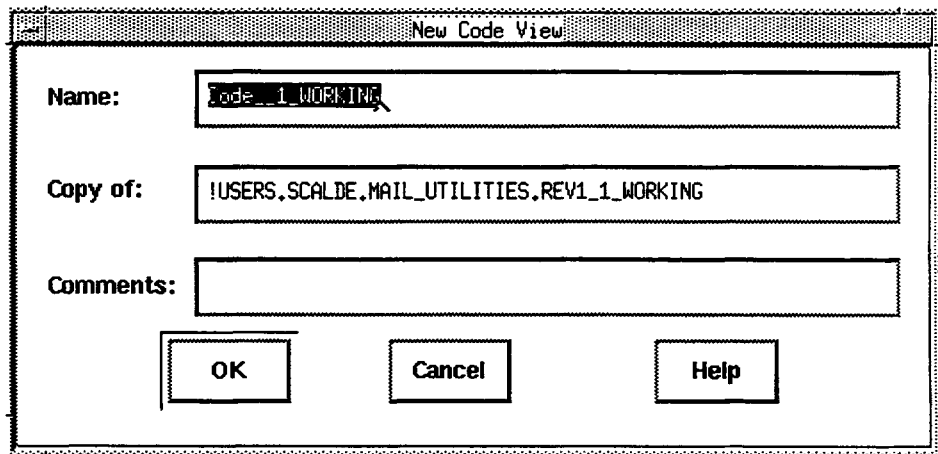
Making a Code View

Code views are copies of views that contain only the executable code from the view. Code views are especially useful when security requirements restrict visibility to portions of source code.

To make a code view:

1. Place the Environment cursor in the view.
2. Choose **File:New:Code View**.

The following dialog box appears:



3. In the **Name** entry box, you can enter a different name than the generated prompt.
4. In the **Copy Of** entry box, enter or edit the name of the source view.
5. In the **Comments** entry box, enter any comments.
6. Click **OK**.

CREATING A SYSTEM

When an application consists of multiple subsystems, these subsystems optionally can be included in an Environment object called a *system*. A system pulls an application's components together by logically grouping particular releases from several component subsystems. Systems:

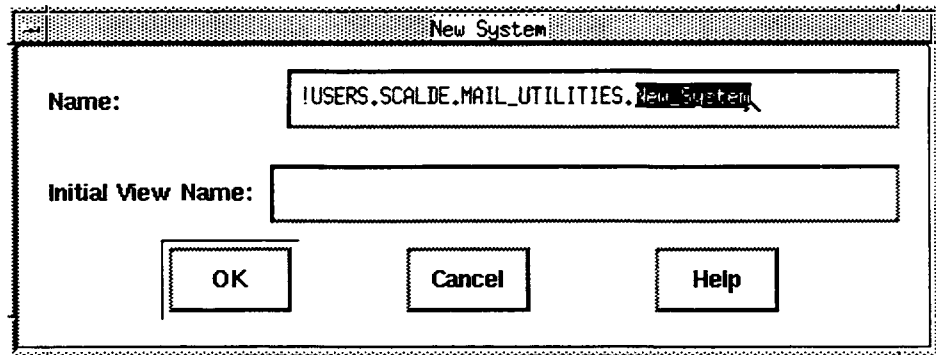
- Provide a way of identifying particular subsystems as components of a given application or of a major portion of an application
- Provide an automated means of tracking the latest release from each subsystem and building activities that reference those releases

- Establish a parent-child relationship between the system and the subsystem
- Have the same internal library structure as subsystems

To create a system:

1. Choose **File:New:System**.

The following dialog box appears:



2. In the Name entry box, you can enter a new name for the system at the prompt.
3. In the Initial View Name entry box, enter a name for the initial view of the system.
4. Click OK.

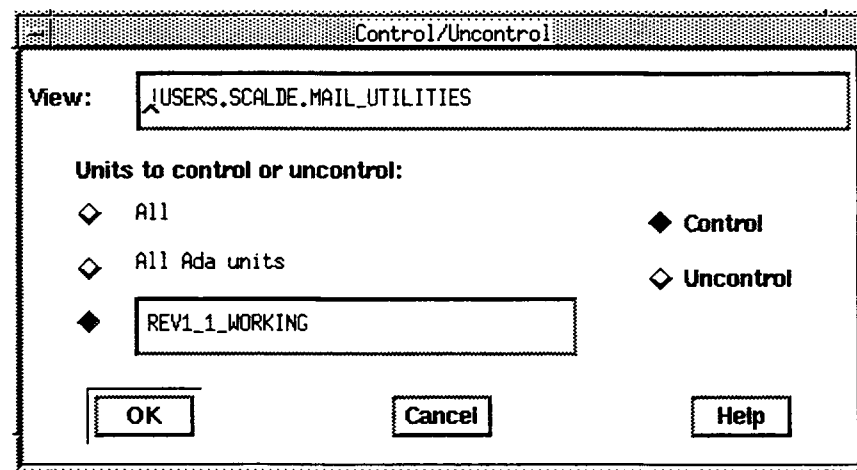
MAKING OBJECTS CONTROLLED OR UNCONTROLLED

When a component of an application is encapsulated in a subsystem, individual objects in the component can be *controlled*—that is, made subject to version control. Controlled objects must be checked out to be modified and, when desired, the modified object can then be checked in and made available for other users to check out.

To change the control status of an individual object:

1. Choose **CMVC:Control/Uncontrol**.

The following dialog box appears:



2. Verify the view name in the View entry box.
3. Click the desired radio button in the Units to control or uncontrol field.

If you choose the third radio button with the entry box, you select the listed Ada unit to be controlled or uncontrolled. You can enter the name of a new Ada unit or units in the entry box or edit the one that is there.

4. Click the Control or Uncontrol radio button to determine the desired state of the unit(s).
5. Click OK.

CHECKING OUT AN OBJECT FOR CHANGES

Controlled objects must be *checked out* to be modified; checking out an object reserves it for editing by acquiring the object's reservation token. When desired, the modified object then can be checked in and made available for other users to check out.

To check out an object:

1. Choose CMVC:Check Out.

The following dialog box appears:

2. Verify the pathname in the Name entry box.
3. Enter any comments in the Comments entry box.
4. Click the desired check boxes.
5. Click OK.

Alternative: Press [Check Out] to check out a unit for changes.

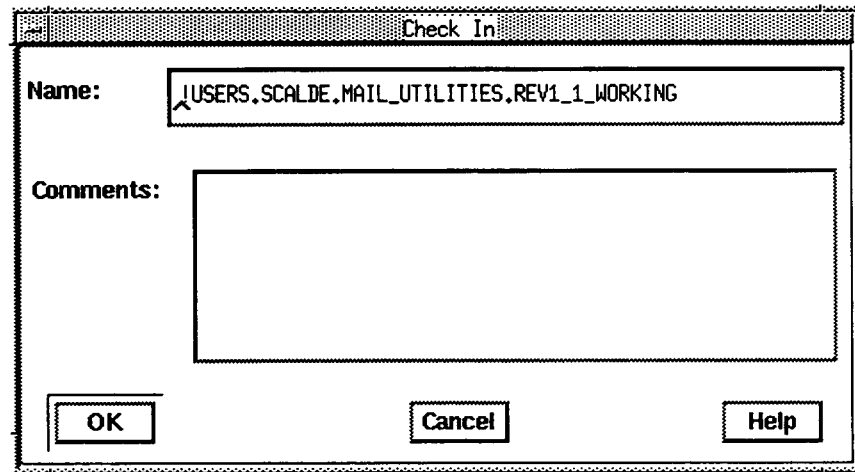
CHECKING IN AN OBJECT AFTER CHANGES

When you have finished modifying a controlled object and you want the changes you made to be recorded in the CMVC database, you must *check in* the object.

To check in an object:

1. Choose CMVC:Check In.

The following dialog box appears:



2. Verify the pathname in the Name entry box.
3. Enter any comments in the Comments entry box.
4. Press OK.

Alternative: Press [Check In] to check in a unit after changes.

ACCEPTING CHANGES

A subpath can become out of date when objects are checked out and modified in other subpaths. Objects in a subpath can be brought up to date by *accepting changes*, usually from the latest generation into that subpath. Accepting changes is useful if you want to:

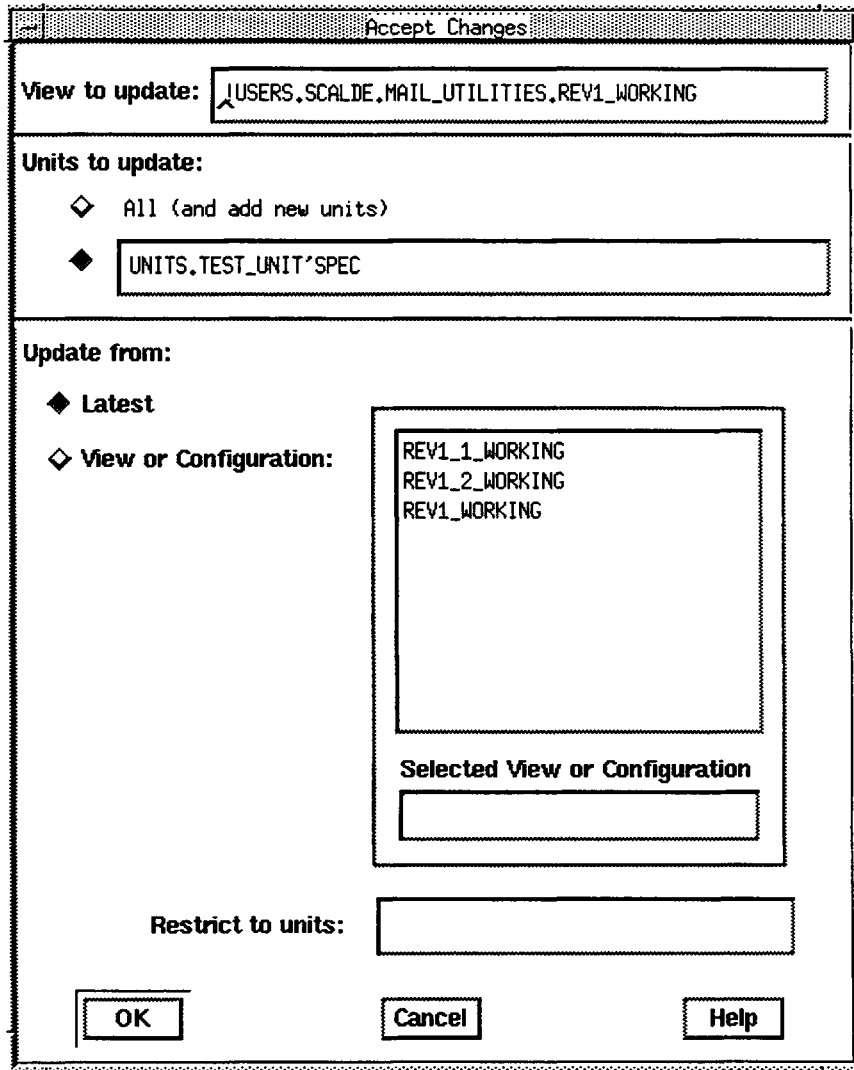
- Synchronize the development of controlled objects that are joined to objects in other views.
- Update out-of-date objects to the latest generation.
- “Go backward in time” to a previous generation of a controlled object that is joined to a less-recently updated object in another view.
- Copy new controlled objects between views.

Accepting Changes from a View

Accepting Changes If the Destination Is an Object

1. Place the Environment cursor in the Units directory of the view into which you want to accept changes.
2. Choose CMVC:Accept Changes.

The following dialog box appears:



3. Verify that the **View to update** entry box contains the name of the view into which you want to accept changes.
4. In the **Units to update** area, choose the objects you want to update:
 - If you want to copy new controlled objects from the source view to the new view, check **All (and add new units)**. You should also check this button if you want to update all objects in the view.
 - If you want to update only certain objects, list those objects in the entry box and check the radio button next to that box.
5. Choose the source from which you would like to accept changes:
 - If you want to accept changes from the latest generation of each controlled object, check **Latest**. If the **View to update** already contains the latest generation of a particular object, that object will not be updated.
 - If you want to accept changes from a particular view or configuration, check **View or Configuration** and select the view or configuration from the list on the right. If the view or configuration from which you want to accept changes is

not in the list, you can enter its name directly in the **Selected View or Configuration** entry box.

6. If you want to restrict the objects from which to accept changes, enter those objects in the **Restrict to units** entry box. Note that this entry box differs from the **Units to update** option in that this specifies the objects *from* which changes will be accepted; **Units to update** controls the objects *into* which changes will be accepted. If the **Restrict to units** entry box is empty, all objects from the specified view or configuration are assumed.
7. Click OK.

Accepting Changes If the Destination Is a View

1. Place the Environment cursor in the view you want to make current.
2. Choose **CMVC:Accept Changes**.
The **Accept Changes** dialog box appears (see above).
3. Click the **All (and add new units)** radio button.
4. Click the **View or Configuration** check box.
5. Click the view you want in the list box.
The name of the view appears in the **Selected View or Configuration** entry box.
6. Click OK.

Note: If you want to update the object to the most recently checked-in generation, click the **Latest** radio button in the **Update from** field.

Accepting Changes from an Object

1. Place the Environment cursor on the object you want to make current.
2. Choose **CMVC:Accept Changes**.
The **Accept Changes** dialog box appears (see above).
3. Click the **All (and add new units)** radio button.
4. Click the **View or Configuration** check box.
5. Click the view that you want in the list box.
The name of the view appears in the **Selected View or Configuration** entry box.
6. Enter the unit(s) you want the update restricted to in the **Restrict to units** entry box.
7. Click OK.

Note: If you want to update the object to the most recently checked-in generation, click the **Latest** radio button in the **Update from** field.

JOINING OBJECTS IN DIFFERENT VIEWS

When a team of developers needs to work on objects in the same subsystem, multiple development subpaths are usually set up. Subpaths usually are created such that the objects in them are *joined* automatically. Joining objects in different subpaths facilitates parallel development because:

- Objects that are joined share a single reservation token. Consequently, a joined object can be checked out in only one view at a time.

- Objects that are joined are represented as a single series of generations stored in the CMVC database. Consequently, changes made to one object can be propagated to the other objects to which it is joined.

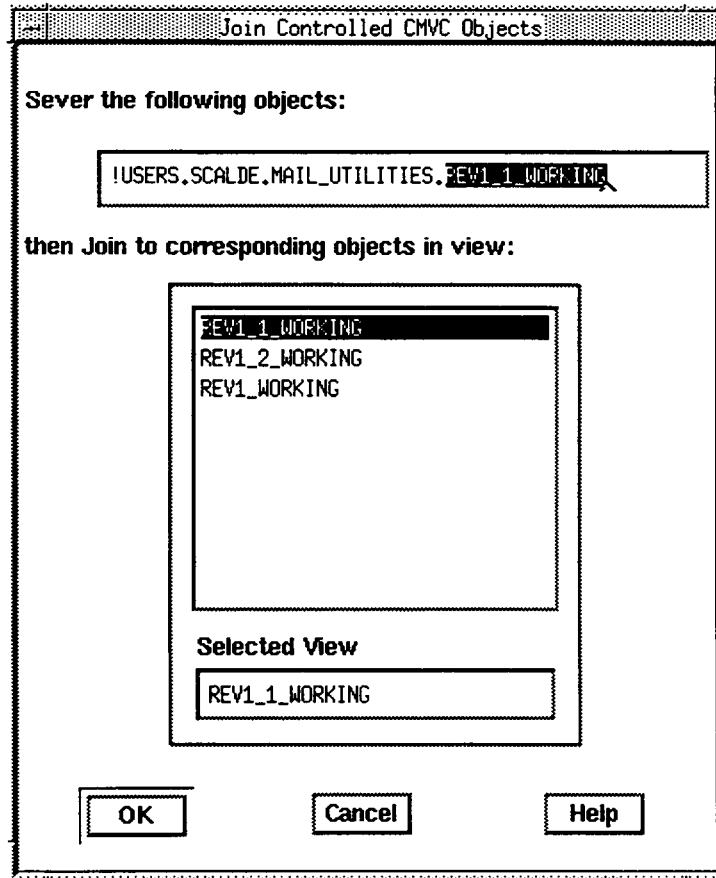
If subpaths are set up such that the objects in them are not joined automatically, you can join two objects provided that:

- The objects have the same pathname within their respective views.
- The objects to be joined are textually identical.

To join two objects:

1. Choose CMVC:Join.

The following dialog box appears:



2. Verify that the object(s) to be severed are listed in the Sever the following objects entry box.
3. Double-click the view you want to join the object(s) to in the list box.
The view is highlighted and displayed in the Selected View entry box.
4. Click OK to join the objects.

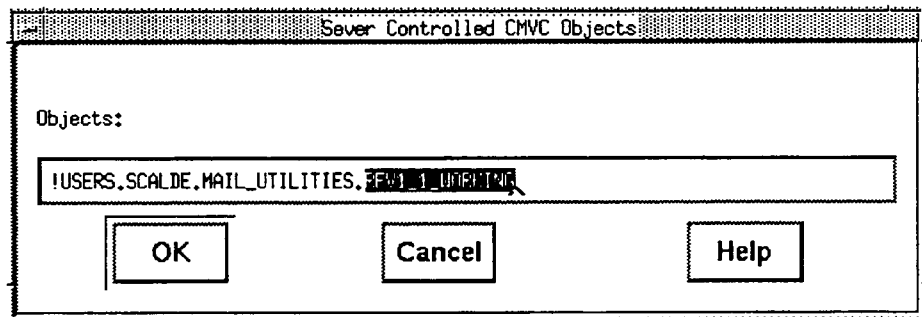
SEVERING OBJECTS IN DIFFERENT VIEWS

If two users need concurrent access to a controlled object, you can *sever* the object between views. Severing provides each copy of the object with its own reservation

token, so that each copy can be checked out independently. Separate sets of generations are kept for severed objects.

1. Choose CMVC:Sever.

The following dialog box appears:



2. Enter the name of the object(s) to be severed. By default, the selected object is named in the entry box.
3. Click OK.

Progress of the procedure is displayed in an I/O window.

REVERTING TO A PREVIOUS GENERATION

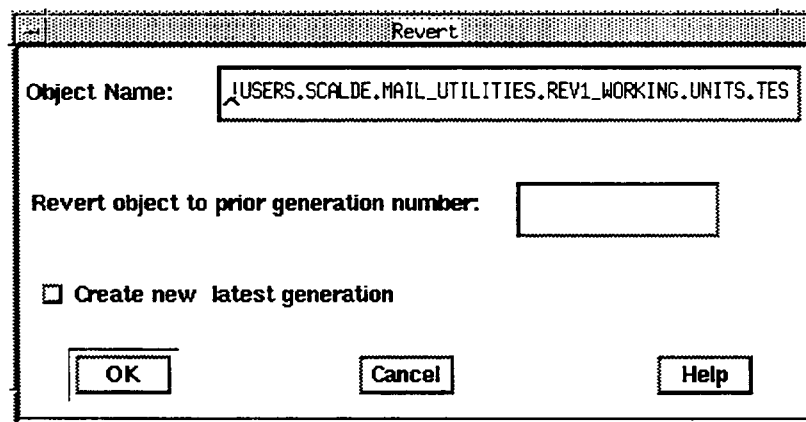
Each time you check out an object, a new *generation* of that object is created in the CMVC database. Editing changes are collected in the new generation and saved in the CMVC database when you check in the object. Thus, generations capture the changes made from checkout to checkout.

Each generation of an object is numbered, starting with generation 1. Generation 1 is created when you make an object controlled; initially generation 1 contains the text of the object at the time it was made controlled. Over time the CMVC database builds up a series of numbered generations for each controlled object.

You can revert an object to any previous generation stored in the CMVC database.

1. Place the Environment cursor on the object you want to revert to a previous generation.
2. Choose CMVC:Revert to Generation.

The following dialog box appears:



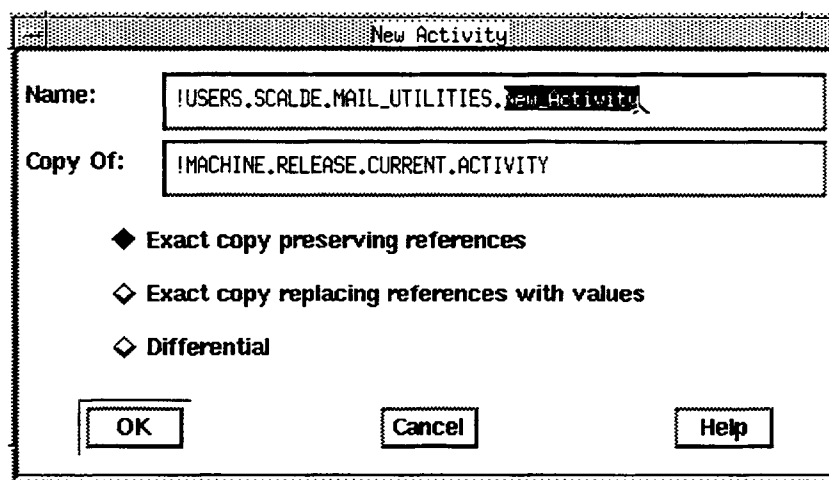
3. Verify the name of the object in the Object Name field.
4. Enter the number of the generation you want to return to in the Revert object to prior generation number entry box.
5. To create a new latest generation, check the Create new latest generation box.
6. Click OK.

CREATING A NEW ACTIVITY

An *activity* is an execution table that must be set up to specify which of the alternative load views is to be used from each subsystem. The activity contains one entry for each subsystem that is required for execution.

1. Choose File:New:Activity.

The following dialog box appears:



2. In the Name entry box, enter the name of the new activity.
3. In the Copy Of entry box, enter the name of the activity from which the new activity is to be created.
4. Click the desired radio button.
To create a new activity with indirect entries for all subsystems in the source activity !Machine.Release.Current.Activity, click the Differential radio button.
5. Click OK.

The new activity is listed in the subsystem directory.

Adding an Activity Entry

1. To view the activity that you have as the default, choose CMVC:Activity.
2. Place the Environment cursor in the activity.
3. Choose Edit:Add Entry.
A command window opens with empty parameters for the names of the subsystem, spec view, and load view.
4. Enter the parameter names.
5. Press [Promote] to add the new activity entry.

STARTING THE CMVC EDITOR

The CMVC editor displays a *configuration image* for the specified view or configuration object or for the view enclosing the specified object. A configuration image for a view is a library-like display of CMVC information pertaining to that view. A configuration image for a view:

- Contains an entry for each controlled object in the view
- Indicates the latest generation that exists for that object in any view

To start the CMVC editor:

1. Place the Environment cursor in the view for which you want information.
2. Choose CMVC:Editor.

Information about the view is displayed in an Environment window. The initial display lists all of the controlled objects in the view and the generation of each. A two-part generation indicates that the specified view does not have the most current generation (for example, 4/8 indicates generation 4 of 8).

From this display, you can obtain more information about each object, such as date and user who last modified it, by pressing [Expand] (generally bound to [Meta][!]). To display the release history for the view, press [Explain] (generally bound to [F3]).

COLLECTING INFORMATION ABOUT CONTROLLED OBJECTS

Information about a CMVC-based project can be gathered in several ways. Each generation of every controlled object has *notes* associated with it, which can be used as a scratchpad for arbitrary commentary. In addition, the date, time, and comments from checkout and checkin operations are automatically logged in an object's notes. These notes are the basis for three kinds of objects that can be associated with user sessions to collect and convey data about the project:

- *Work orders* are designed to communicate details about specific tasks to be accomplished. When development proceeds in response to a given work order, time-stamped comments are logged to the work order whenever any command from package CMVC (or from the CMVC menu) is executed.
- *Work-order lists* are composed of a group of work orders.
- *Ventures* contain information about groups of work orders and work-order lists and control their use. Each work order must have a venture that is its parent.

For complete information about using work orders, work-order lists, and ventures, see the Project Management (PM) book of the *Rational Environment Reference Manual*.

Creating a Work Order

Work orders are designed to communicate details about specific tasks to be accomplished.

To create a work order:

1. Choose File:New:Work Order.

The following dialog box appears:

2. In the **Name** entry box, enter the name of the work order.
3. In the **Notes** entry box, enter notes about the work order.
4. In the **Venture** entry box, enter the name of the work order venture (see "Creating a Venture," below).
5. In the **Work Order List** entry box, enter the name of the work-order list (see "Creating a Work-Order List," below).
6. Click **OK**.
Progress of the procedure is displayed in an I/O window.

Creating a Work-Order List

Groups of related work orders constitute a *work-order list*. For example, a work-order list may relate to a particular module of code or it may be the set of work orders assigned to an individual developer.

To create a work-order list:

1. Choose **File:New:Work Order List**.

The following dialog box appears:

2. In the Name entry box, enter the name of the new work-order list.
3. In the Ventures entry box, enter the name of the parent venture.
4. To make the new list become the new default work-order list in the specified venture, click the **Make Default** check box.
5. Click **OK**.

Progress of the procedure is displayed in an I/O window.

Creating a Venture

A *venture* is a management tool that contains information about groups of work orders and work-order lists and controls their use. Each work order must have a venture that is its parent.

1. Choose **File:New:Venture**.

The following dialog box appears:

2. In the Name entry box, enter the name of the new venture.
3. In the Notes entry box, enter any notes or comments.
4. To make this venture the default venture for the current session, click the **Make Default** check box.
5. Click **OK**.

Progress of the procedure is displayed in an I/O window.

GETTING INFORMATION ABOUT A VIEW

Table 12-1 shows the commands in the CMVC:CMVC Report submenu that list information in an I/O window.

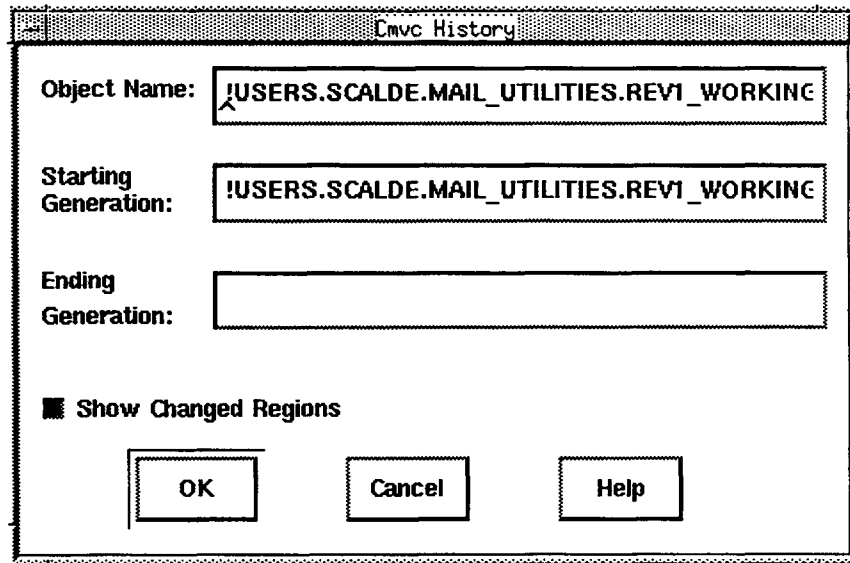
Table 12-1 CMVC Report Commands

Command	Lists
CMVC:CMVC Report:List Controlled	Controlled objects
CMVC:CMVC Report:List Uncontrolled	Uncontrolled objects
CMVC:CMVC Report:List Out-of-Date Objects	Out-of-date objects
CMVC:CMVC Report:List Checked-Out Objects	Checked-out objects
CMVC:CMVC Report:List View Imports	View imports
CMVC:CMVC Report:List View Exports	View exports
CMVC:CMVC Report:List View Referencers	View referencers
CMVC:CMVC Report:List View Model	View model

Getting the History of an Object

1. Choose CMVC:CMVC Report:Object History Information.

The following dialog box appears:



2. Verify the pathnames of the object and its starting generation in the Object Name and Starting Generation entry boxes.
3. In the Ending Generation entry box, enter the pathname of the last generation whose history you want to see.

4. To see the changes that have occurred between the starting and ending generations you specified, select the **Show Changed Regions** entry box.

If this box is unselected, you will see only check in/out information and comments.

5. Click OK.

A log appears in an I/O window with information about the object.

13

Controlling Jobs

A *job* consists of one or more commands that are executed together. A job is initiated each time you edit an object or execute a command, using the menus, the mouse, a key combination, or a command window. Jobs also can be initiated programmatically from other jobs using commands in package !Commands.Program (see the Session and Job Management (SJM) book of the *Rational Environment Reference Manual*).

When a job is created, the Environment assigns it a *job identification number* (*job ID*). The job ID is often required by commands that connect, disable, enable, or kill jobs.

This chapter describes how to display, disconnect from, disable, enable, and kill jobs using Access. Normally, these operations can be performed only on jobs associated with your username. To manipulate a job that belongs to another user, you must have operator capability. (For information about operator capability, see package Operator in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.)

DISPLAYING CURRENT JOBS

Displaying Your Current Jobs

Choose `Session:Jobs:Show Jobs`.

The I/O window appears with a list of your current jobs:

- `Line` indicates the port number through which you are logged in.
- `Job` indicates the job identification number. You need to know the job number if you want to connect, disable, enable, or kill a job.
- `S` indicates the job state. Job states include the following:
 - `RUN` indicates that the job is consuming CPU time or is eligible to consume CPU time.
 - `IDLE` indicates that the job is not consuming CPU time and has no unblocked tasks. An idle job may be waiting for input or requests for service.
 - `WAIT` indicates that the job is able to run but is temporarily ineligible for CPU time.
 - `DISABLED` indicates that the job has been explicitly disabled by a user.
 - `QUEUED` indicates that the job is waiting to run in one of the background queues.

- **Time** indicates the elapsed time since the job began. The time for the editor and command jobs indicates how long you have been logged in.
- **Job Name** identifies the command or commands that are being executed as part of the job. The editor and command jobs represent your session.

Displaying All Current Jobs

Choose **Session:Jobs:Show All Jobs**.

The I/O window appears with a log of all current user activity:

- **User** indicates the username under which the job is running. The username *SYSTEM indicates jobs that belong to the Environment itself.
- **Line** indicates the port number through which the user is logged in. A dash (-) indicates that the user is not currently logged into the Environment.
- **Job** indicates the job identification number. You need to know the job number if you want to connect, disable, enable, or kill a job.
- **S** indicates the job state. Job states include the following:
 - **RUN** indicates that the job is consuming CPU time or is eligible to consume CPU time.
 - **IDLE** indicates that the job is not consuming CPU time and has no unblocked tasks. An idle job may be waiting for input or requests for service.
 - **WAIT** indicates that the job is able to run but is temporarily ineligible for CPU time.
 - **DISABLED** indicates that the job has been explicitly disabled by a user.
 - **QUEUED** indicates that the job is waiting to run in one of the background queues.
- **Time** indicates the elapsed time since the job began. The time for the editor and command jobs indicates how long you have been logged in.
- **Job Name** identifies the command or commands that are being executed as part of the job. The editor and command jobs represent your session.

DISCONNECTING FROM A JOB (PUTTING IT IN THE BACKGROUND)

Normally, user-initiated jobs are run in the foreground as *connected* (or *attached*) jobs. However, you can cause user-initiated jobs to execute in the background, making them *disconnected* (or *detached*) jobs. Disconnecting from a job allows you to enter other commands while the job continues executing.

To disconnect from the current job:

- Press [Control][G].

A user-interrupt message is displayed in the message window. You can now move the cursor and perform other tasks. The job continues to execute. Note, however, that background jobs are allocated the remaining resources after foreground jobs are handled; thus, they may execute more slowly.

Note: There is a *Disconnect Current Job* option on the *Session* menu. This command is not functional from the menu, however, because menu operations cannot be executed

while a job is running. The entry is only a reminder that [Control][G] will disconnect a job.

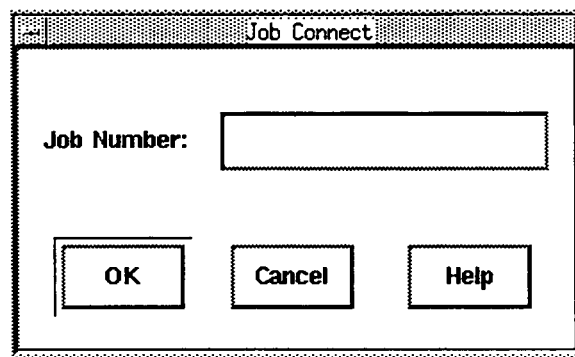
RECONNECTING TO A JOB (PUTTING IT IN THE FOREGROUND)

If you have disconnected from a job, you can reconnect to it, making it a *connected* (or *attached*) job and bringing it into the foreground. A connected job maintains control of your session, preventing you from entering other commands. However, connected jobs are allocated the majority of system resources and, thus, execute more quickly than disconnected jobs.

To connect to a previously disconnected job:

1. Choose **Session:Jobs:Connect to Job**.

The following dialog box appears:



2. Enter the number of the job to which you want to reconnect in the **Job Number** entry box.
3. Click **OK**.

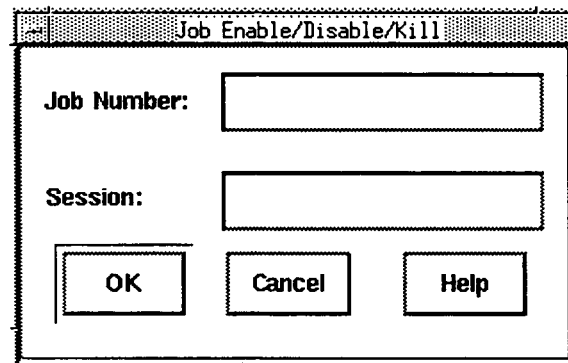
Your session is reconnected to the specified job and the `...running` message appears in the banner of the message window.

DISABLING A JOB

To temporarily stop a job from executing without terminating the job:

1. Choose **Session:Jobs:Disable Job**.

The following dialog box appears:



2. Enter the number of the job you want to disable in the **Job Number** entry box.
3. Enter the name of the session under which the job is running in the **Session** entry box.
4. Click **OK**.

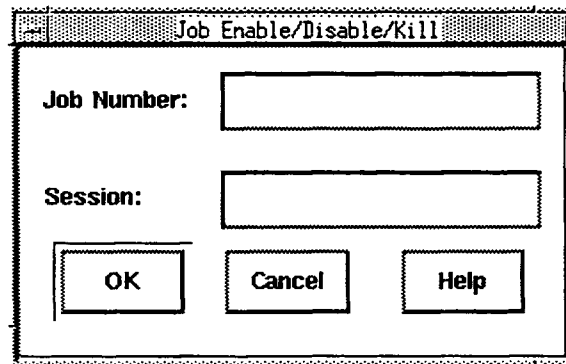
Note: To find a list of job numbers, see "Displaying Current Jobs," above.

ENABLING A JOB

To restart a job that has been explicitly disabled:

1. Choose **Session:Jobs:Enable Job**.

The following dialog box appears:



2. Enter the number of the job to be enabled in the **Job Number** entry box.
3. Enter the name of the session under which the job is disabled in the **Session** entry box.
4. Click **OK**.

Note: To find a list of job numbers, see "Displaying Current Jobs," above.

KILLING A JOB

Killing the Current Job or the Last Job Created

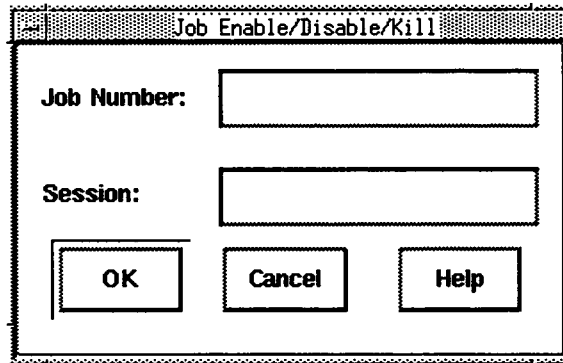
To kill the current job or last job created, press **[Meta][G]**.

The job is terminated, and a job-abort message is displayed in the message window. All files declared as **Current_Output**, **Current_Error**, or **Current_Input**, or files on the **current-output**, **current-error**, or **current-input** stacks, are closed. All other files are abandoned, and temporary files are deleted.

Killing Any Job

1. Choose **Session:Jobs:Kill Job**.

The following dialog box appears:



2. Enter the number of the job to be killed in the **Job Number** entry box.
3. Enter the name of the session under which the job is running in the **Session** entry box.
4. Click **OK**.

The job is terminated, and a job-abort message is displayed in the message window. All files declared as `Current_Output`, `Current_Error`, or `Current_Input`, or files on the current-output, current-error, or current-input stacks, are closed. All other files are abandoned, and temporary files are deleted.

14

Customizing Your Access Workspace

This chapter describes how to create and execute user-defined buttons and keyboard macros, redefine keys, and manipulate the sash control on a main Access window.

EXECUTING MENU COMMANDS WITH USER-DEFINED BUTTONS

Depending on what menu commands you use often, you may want to create your own user-defined buttons (see Figure 14-1). These buttons allow you to execute a menu command by clicking on the button.

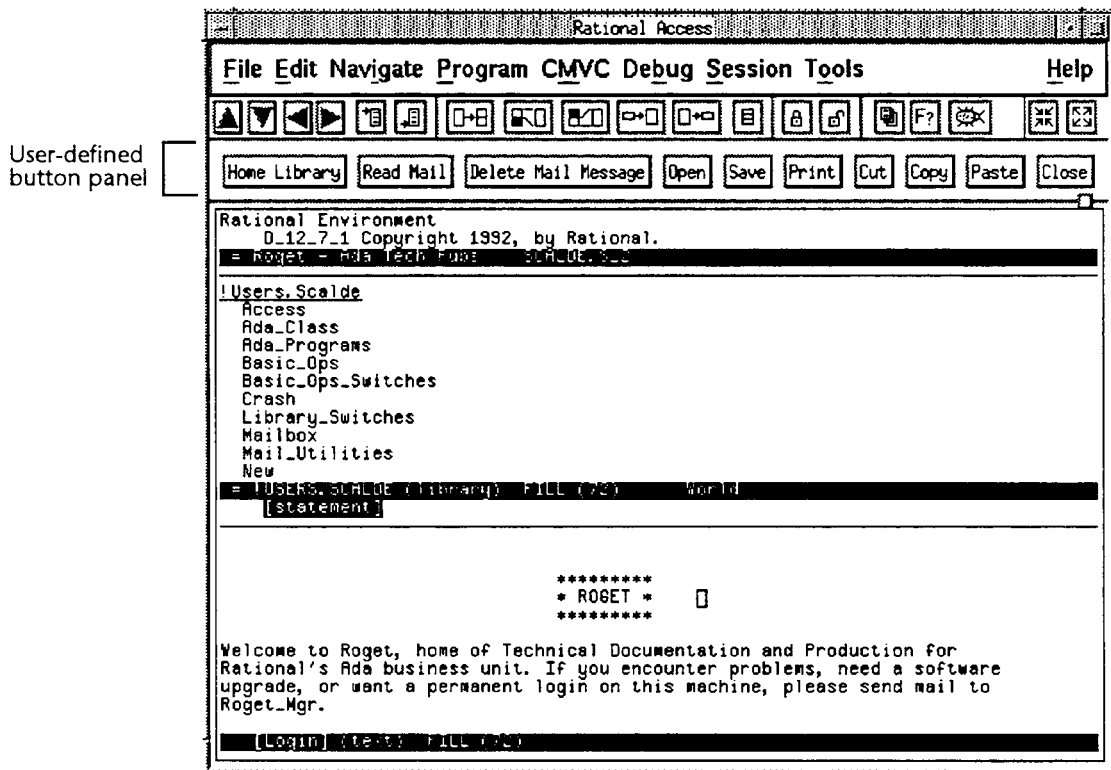


Figure 14-1 User-Defined Button Panel

The user-defined buttons are displayed as the name of the command enclosed in a rectangular border, and they appear directly below the window-control buttons and above the sash. You can create as many buttons as you wish, making space for them by pulling down the sash in the Environment area. The buttons appear left to right, in the order in which you create them. A button can be deleted at any time; remaining buttons are relocated to maintain packing. You can save the set of defined but-

tons at any time. The last saved set of buttons is presented automatically when you create future main Access windows.

Note: If your Access window is as tall as the screen allows, creating a user button that starts a new row (for example, the very first button you create) has no visible effect; you have to pull down the sash manually to see the button.

Creating a Button for a Menu Command

1. Move the pointer to the desired menu or submenu command.
2. Press [Control] and click the left mouse button.

A button with the name of the command appears above the sash.


Buttons appear from left to right above the sash in the order they are created. You can create multiple rows of buttons.

Note: You cannot create buttons for File:Exit, for the Session:Screen toggle switches, for Help menu items except for Help:Explain Underline and Help:On Key, or for commands on the Function Key Palette.

Changing the Size of the Button Area

You can move the sash that separates the Environment area from the user-defined button panel.

To move the sash:

1. Place the pointer on the sash control: 
The pointer becomes a cross.
2. Drag the pointer up or down to move the sash.

Activating a User-Defined Button

Place the pointer on the button and click.

Deleting Buttons

1. Place the pointer on the button.
2. Press [Control] and click the left mouse button.

The button disappears.

Saving Buttons

Choose Session:Screen:Save Button Panel.

Your current button panel is saved and will reappear whenever you create a main Access window.

BUILDING AND EXECUTING MACROS

Defining a Macro

1. Choose **Tools:Macro:Begin Macro Def.**

The message **Defining keyboard macro** appears in the message window.

2. Press the key sequence that you want to make into a macro.
3. Choose **Tools:Macro:End Macro Def.**

The message **End of keyboard macro** appears in the message window.

The macro is now defined.

***Note:** You cannot make mouse operations part of a macro.*

***Alternative:** Press [Meta][] to begin macro definition and [Meta][] to end macro definition.*

Executing a Macro

Choose **Tools:Macro:Execute Macro.**

The macro is executed.

***Alternative:** Press [Meta][X] to execute the macro.*

Binding a Macro to a Key

Before starting, you may want to execute the **Help: On Key** operation to see if the key is already bound.

1. Choose **Tools:Macro:Bind Macro to Key.**

A message appears in the message window prompting you for the key that will bind the macro.

2. Press the key to be bound.

You can now execute the macro by pressing the key.

***Alternative:** Press [Control][Meta][=] or [Control][Meta][+] to bind a macro to a key.*

Saving the Current Macros

1. Press [Cmd Window] to create a command window.
2. Enter **Macro.Save.**
3. Press [Promote].

All macros currently bound to keys are permanently saved.

REBINDING KEYS

Before starting, you may want to execute the Help: On Key operation to see if the key is already bound.

Rebinding Temporarily

1. Press [Cmd Window] to create a command window.
2. Enter `Key.Define`.
3. Press [Complete].
4. At the `Key_Name` prompt, enter the key you want to rebind to the new command.
If you do not know the name of the key, choose `Help:On Key` and then press the key for which you want to know the name. The key name for that key is displayed in the message window.
5. At the `Command_Name` prompt, enter the name of the command you want bound to this key.
6. Press [Promote].

Rebinding Permanently

1. Place the Environment cursor in your home world.
2. Create a procedure named `Rational_Access_Commands` by copying the text from the template in `!Machine.Editor_Data.Rational_Access_User_Commands` into an Ada unit.
See the section titled "Creating an Ada Unit" in Chapter 7 for details.
3. Edit the body of `Rational_Commands` so that the case statement contains alternatives for those keys you want to rebind.
4. Choose `Program:Promote to Installed`.

The changes will be in effect when you next log in.

For more information about rebinding keys, see the Key Concepts section of the Session and Job Management (SJM) book of the *Rational Environment Reference Manual*.

CHANGING THE SCREEN TO INVERSE VIDEO

Choose `Session:Screen:Inverse Video` to toggle the inverse video on or off.

The screen inverts the text color with the Access area background color.

SETTING THE VISUAL BELL

Choose `Session:Screen:Visual Bell` to toggle the bell on or off.

When the visual bell is on, the screen will flash once to indicate an error instead of beeping.

15

Using Command Windows

Certain operations are not available from Access menus and buttons. To perform these, you execute Environment commands from command windows. Command windows are actually small Ada programs; Environment commands are Ada procedures. Command windows are also useful for performing several commands in a single operation and for executing your own commands. Note that all Environment commands and most Access commands can be executed from a command window. See Appendix C for a list of the Environment commands from which Access commands are built.

To find out which Environment commands are most closely related to a particular menu item, see the online help for that item. If you want to know the menu item or button that is most closely related to an Environment command, see Appendix C, "Access Equivalents: Environment Commands."

CREATING AND EXECUTING A COMMAND-WINDOW PROGRAM

1. Press [Cmd Window] to open a command window.
2. Enter the Environment command, formatting frequently for multiple-line programs by pressing [Format].
3. Press [Semanticize] to check for semantic errors.
The Environment underlines existing errors.
4. Correct any errors and semanticize again.
5. Press [Promote] to execute the program.

***Alternative:** Press [Meta][C] to create a command window. Choose File:Run or Program:Promote to execute the program.*

GETTING COMMAND COMPLETION

1. Place the Environment cursor in a command window.
2. Enter some fragment of a command name.
 - You can supply only a command name or name fragment. Completion will fail if you enter any part of the argument list, including the parenthesis that begins the list.
 - Completion ignores final semicolons if any exist (for example, if you have used the Format command and it added a semicolon after the name or name fragment).
3. Press [Complete].

This completes the command and provides prompting for any parameters.

***Note:** If the command fragment is ambiguous, the complete operation fails and the Environment displays the possibilities in another window. Select the desired command from the display ([Control]+click), and press [Complete] again. The Environment will fill in the new command in the command window.*

***Alternative:** To get completion, choose Program:Complete.*

MOVING IN A COMMAND WINDOW

Moving to an Underline

To move to the next underline, choose `Navigate:Next Underline`.

To move to the previous underline, choose `Navigate:Previous Underline`.

***Alternative:** To move to the next underline, press [Meta][=] or [Meta][+], or press [Meta][-] or [Meta][_] to move to the previous underline.*

Moving to a Prompt or Underline

To move to the next prompt (in reverse video) or underline, choose `Navigate:Next Item`.

To move to the previous prompt or underline, choose `Navigate:Previous Item`.

***Alternative:** Press [Next] or [Meta][↑] to move to the previous prompt or underline, or [Previous] or [Meta][↓] to move to the next prompt or underline.*

Turning Off a Prompt

1. Place the Environment cursor on the prompt to be turned off.
2. Choose `Edit:Deselect`.

The prompt becomes normal video, and the text can now be edited.

***Alternative:** Press [Control][X].*

Turning Off Underlines

1. Place the Environment cursor anywhere in the window.
2. Press the [Und Off] key.

All underlines are removed from the current window.

***Alternative:** Press [Control][X].*

REEXECUTING A COMMAND

1. Place the Environment cursor in the command window containing the command to be reexecuted.
2. Press [Promote].

The command is executed again.

ENTERING A NEW COMMAND

Entering a New Command in the Same Window

1. Delete the command.
 - If the old command is not in reverse video, clear the command of text (see below).
 - If the old command is in reverse video, place the Environment cursor on the command.
2. Enter the new command.

If the old command was in reverse video, it will disappear and be replaced by the new command.
3. Press [Promote] to execute the new command.

Clearing a Command Window of Unneeded Text

1. Place the Environment cursor in the command window to be cleared.
2. Choose File:Revert.

A [statement] prompt appears in place of the text. You can enter a new command.

Alternative: Press [Edit].

GOING BACK TO PREVIOUS COMMANDS

The Environment maintains a history of commands and Ada programs for each command window. You can access and execute any of the commands in this sequential history.

Redisplaying a Previous Command in the Historical Sequence

1. Place the Environment cursor in the command window.
2. Press [Control][Meta][<].

The previous command entered in the command window appears.

Repeat the sequence to see earlier commands.

Redisplaying a Later Command in the Historical Sequence

1. Place the Environment cursor in the command window.
2. Press [Control][Meta][>].

The next command in the historical sequence appears.

Repeat the steps to see later commands in the sequence.

GETTING THE PARAMETERS OF A COMMAND BOUND TO A KEY

1. Press [Prompt For].
A Prompt For: message appears in the message window.
2. Press the key or key combination whose command parameters you want to see.

The command bound to the key you pressed is listed in the message window, and its parameters appear in the command window.

A

Setting Up Access

This appendix:

- Provides a brief overview of Access components and how they interact
- Describes hardware and software required for running Access and provides sample machine configurations

HOW ACCESS WORKS

Access runs as an X Window System (X) application on a workstation. As such, Access has several software components, which interact with X software components and with the Rational Environment. The following subsections briefly define several X concepts and then describe Access in terms of these concepts.

X Application Components

X is a network-based graphics window system that allows you to work with multiple software applications simultaneously, each in a separate *X window*. Software applications that receive input and/or display output through X windows are called *X clients*. **xclock** is a frequently encountered X client, as is one of Access's two components.

Whereas an X client performs application-specific tasks, the actual communication between the X client and an X window is mediated by a software component called an *X server*. That is, the X server controls the hardware *display*, which consists of one or more screens, plus the associated input devices (generally a keyboard and a mouse or other pointing device). When you enter input using a keyboard or mouse, the X server conveys the input to the appropriate X client. Conversely, when the X client has output, it requests that the X server pass this information to the screen.

One X server is associated with each display. An X server may run on the same workstation that runs the X client, in which case the X client is said to use a *local display*. Alternatively, the X server may run on a different processor on the same network. For example, the X server may run on a special-purpose display called an X terminal or it may run on another workstation, controlling that workstation's display. In this case, the X client is said to use a *remote display*.

A special kind of X client called a *window manager* gives users control over the size and location of each X window, independent of the X client that controls the window's contents. For more information on X, see V. Quercia and T. O'Reilly's *X Window System User's Guide, OSF/Motif Edition* (Sebastopol, CA: O'Reilly & Associates, 1991).

To summarize:

- *X client*: A software application, such as **xclock**, Motif Window Manager (**mwm**), or Access, which communicates with users through one or more X windows on a display.
- *Display*: The screen, keyboard, and mouse you use to communicate with applications in X windows.
- *X server*: The software that mediates between a display and an X client. An X server controls each display and passes information between X clients. An X server can run on a variety of hardware. Two examples are a workstation and a special-purpose display called an *X terminal*.

Access as an X Application

Access has two components:

- *Access X Client*: The portion of Access that runs as an X client on the workstation and that provides the elements of the graphical user interface (menus, command buttons, dialog boxes, and so on). The Access X Client is also a client of (passes information to and from) the Access Server.
- *Access Server*: The portion of Access that runs as a background Environment job on the Rational R1000 Software Engineering Server (the R1000 hardware). The Access Server responds to information from the Access X Client by executing commands in the Environment.

When you type characters (or use the mouse to click) in an Access window:

1. The X server directs the characters/pointer actions to the Access X Client.
2. The Access X Client forwards the characters/pointer actions to the Access Server.
3. The Access Server uses the keymap to translate characters/pointer actions into Environment commands.
4. The Access Server then invokes the appropriate Environment commands and passes information about the command's output to the Access X Client.
5. The Access X Client requests that the X server display any user-visible changes to the screen on your display.

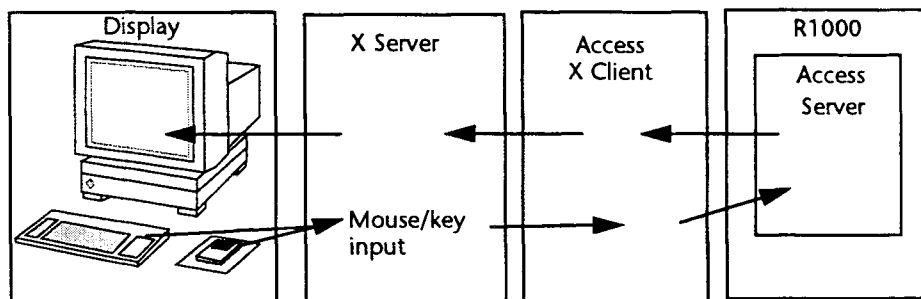


Figure A-1 Sample Information Flow in a Standard Access Setup

REQUIREMENTS FOR RUNNING ACCESS

Each Access user needs:

- A login on a UNIX workstation running a window manager such as the Motif Window Manager (**mwm**). This workstation must also be able to run the Access X Client software. (See the Rational Access Release Information for a list of supported workstations.)
- A display running an X server. This display can be:
 - A local display (that is, a screen connected directly to the workstation that runs the Access X Client).
 - A remote display (that is, a screen associated with some other processor networked to the Access workstation). A remote display may be an X terminal or a screen connected to any workstation that runs an X server.
- An Environment account on the R1000 running the Access Server. If you are not sure whether the Access Server is running, ask your system manager; alternatively, you can:
 1. Enter the `WhatUsers` command from a non-Access interface to the R1000 (for example, from an RXI window).
 2. Look for the name of the Access Server. It will look something like:
`Rational_Access Commands Rev#`
 3. If the job is not listed, have your system manager start the job or install Access if necessary.

Configurations for Using Access

The elements listed above can be configured in several ways. The two most typical configurations are described in the following examples.

Example 1. In the most typical setup, the Access X Client, the X server, and the window manager all run on the same UNIX workstation, as shown in Figure A-2. Access windows are displayed on the workstation's local display. The Access X Client connects to the Access Server on the R1000 through Telnet.

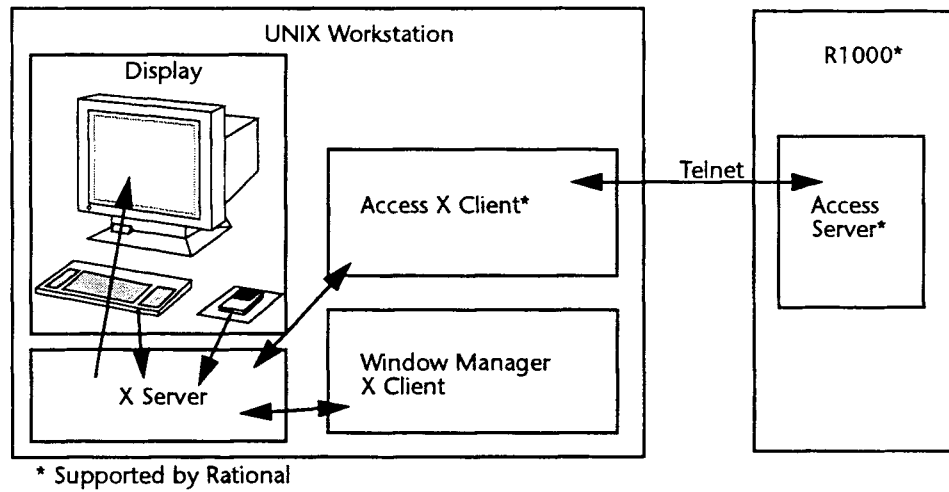


Figure A-2 A Typical Configuration with a UNIX Workstation and a Local Display

Example 2. In an alternative setup, shown in Figure A-3, the Access X Client and the window manager run on a UNIX workstation, while the Access window is displayed remotely on an X terminal (a processor that has only an X server running on it). Note that one workstation can interact with numerous X terminals. As before, the Access X Client connects to the Access Server on the R1000 through Telnet.

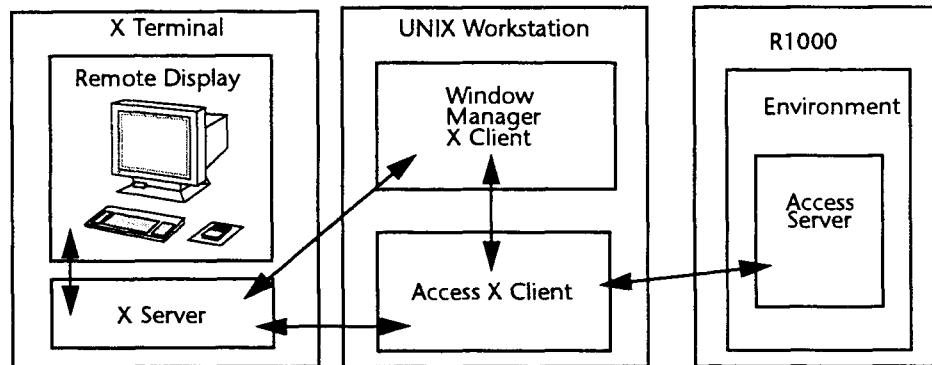


Figure A-3 A Configuration with a UNIX Workstation and a Remote Display

B

User-Interface Basics

Rational Access is an OSF/Motif-based application, which means that it follows OSF/Motif standards for mouse, menu, and dialog-box operations. This appendix provides a brief overview of basic terms and operations for users who are inexperienced with this type of user interface. For more detailed information, see the *OSF/Motif User's Guide*.

CHOOSING MENU COMMANDS

Rational Access commands are available from menus you pull down from the *menu bar* at the top of an Access window.

Terms for Describing Menus

This manual uses the notation `Menu:Command` to refer to commands on menus. For example, `Edit:Copy` refers to the `Copy` command on the `Edit` menu.

Some commands are executed from submenus. Notation for these commands is `Menu:Submenu:Command`. For example, `File:New:Text File` refers to the `Text File` command on the `New` submenu of the `File` menu.

As described in the following sections, you can use the mouse or the keyboard to pull down menus from the menu bar and then choose Access commands from them. Note that:

- When you choose a command whose name is followed by an ellipsis (...), a dialog box appears requesting further information.
- When you choose a command whose name is followed by an arrow, a submenu appears with more command choices.
- If a command name appears in gray, it is currently inapplicable and cannot be chosen.

Using the Mouse to Choose Commands from Menus

There are two ways to choose a command from a menu: clicking with the mouse and dragging with the mouse.

Clicking with the Mouse

1. Put the pointer on the appropriate menu title in the menu bar and click. This opens the menu.

2. Put the pointer on the name of the desired command and click. This initiates the command.

If the command is followed by an arrow, a submenu will appear when you click on the command.

Put the pointer on the name of the desired command in the submenu and click. This initiates the command.

Dragging with the Mouse

1. Put the pointer on the appropriate menu title; press and hold the left mouse button to pull down the menu.
2. Drag the pointer to the desired command (a shaded rectangle appears around the command designated by the pointer).
3. Release the button to initiate the command.

If you drag on a command with a submenu, the submenu appears; it disappears when you pass the command.

Drag to the right to the desired command on the submenu and release to initiate the command.

Table B-1 Summary of Mouse Operations in Menus

Mouse Operation	Result
Click on a menu name from the menu bar	Pulls down the menu
Click on a command name from a menu	Initiates the chosen command from the chosen menu
Drag on a menu, releasing on a command name	Initiates the chosen command from the chosen menu

Using the Keyboard to Choose Commands from Menus

You can use keyboard alternatives to choose menu items, or you can move the keyboard focus to the menu bar and manually traverse through the menu structure.

Using Mnemonics

You can use *mnemonics* to pull down menus and choose commands without using the mouse. A mnemonic is a letter that you can type from the keyboard to activate a particular menu or command. The mnemonic for a given menu or command is the underlined character in the menu title or command name.

You use a modifier key with the mnemonic for a menu to distinguish the mnemonic from other characters you type. The modifier key is indicated in this manual by the logical key name [Meta].

For most users, the [Meta] key is mapped to a physical key labeled [Alt] or [⌘]. If you have customized your key bindings, keep in mind the key you would actually press. Note that the X Window System refers to this key as Meta or Mod1, and OSF/Motif uses **MAlt**.

To use mnemonics to choose a command from a menu:

1. Hold down [Meta] and then press the letter that is underlined in the menu's title; release both keys. This pulls down the menu.
2. While the menu is displayed, type the letter that is underlined in the command's name (you do not need to hold down [Meta]). This initiates the command.

Note that some menu commands have a keyboard shortcut listed next to the menu command name (for example, `Session:Jobs:Kill Job` has a `Meta+G` command-key entry next to the menu command). You can execute the menu command from the keyboard by pressing the designated keys without opening the menu.

Choosing Directly from the Menu Bar

1. Press [Menu Bar].
The keyboard focus moves the File menu, which becomes surrounded by the location cursor (a shaded rectangle).
2. Use [←] and [→] to move the location cursor to the desired menu.
3. Use [↓] to open the menu and reveal its list of commands.
4. Use [↑] and [↓] to move to the command you want to execute. If the command has a submenu (designated by an arrow next to the command), press [→], [Space Bar], or [Return] to open the submenu.
5. To execute the command, press [Space Bar] or [Return].

Executing Window-Control Button Commands

You can execute window-control operations from the buttons on the window-control panel, located below the menu headings.

To execute a button command, put the pointer on the button and click.

For information on the window-control button panel, see Chapter 1, "Getting Started."

Alternative: *You can move the keyboard focus to the window-control button panel by pressing [Control][Tab]. To execute a button command, use the arrow keys to move the location cursor (it appears as a shaded box around the button) to the button and press [Return] or [Space Bar].*

Pressing [Tab] or [Control][Tab] will move the keyboard focus to the user-defined button panel, and repeating will move it to the sash and then back to the Environment area. Note that you cannot move the sash with the keyboard.

RESPONDING TO DIALOG BOXES

Dialog boxes are windows that appear when commands require additional information. Dialog boxes may appear right after you choose a command from a menu, or they may appear during command execution to ask you a question, inform you of an error, or warn you about a potential loss of data.

Note: *Some of the operations described in this section may be affected by the keyboard-focus policy set in your X resources. Contact your UNIX system administrator for more information on X resources.*

Terms for Describing Dialog Boxes

Dialog boxes can contain various types of elements:

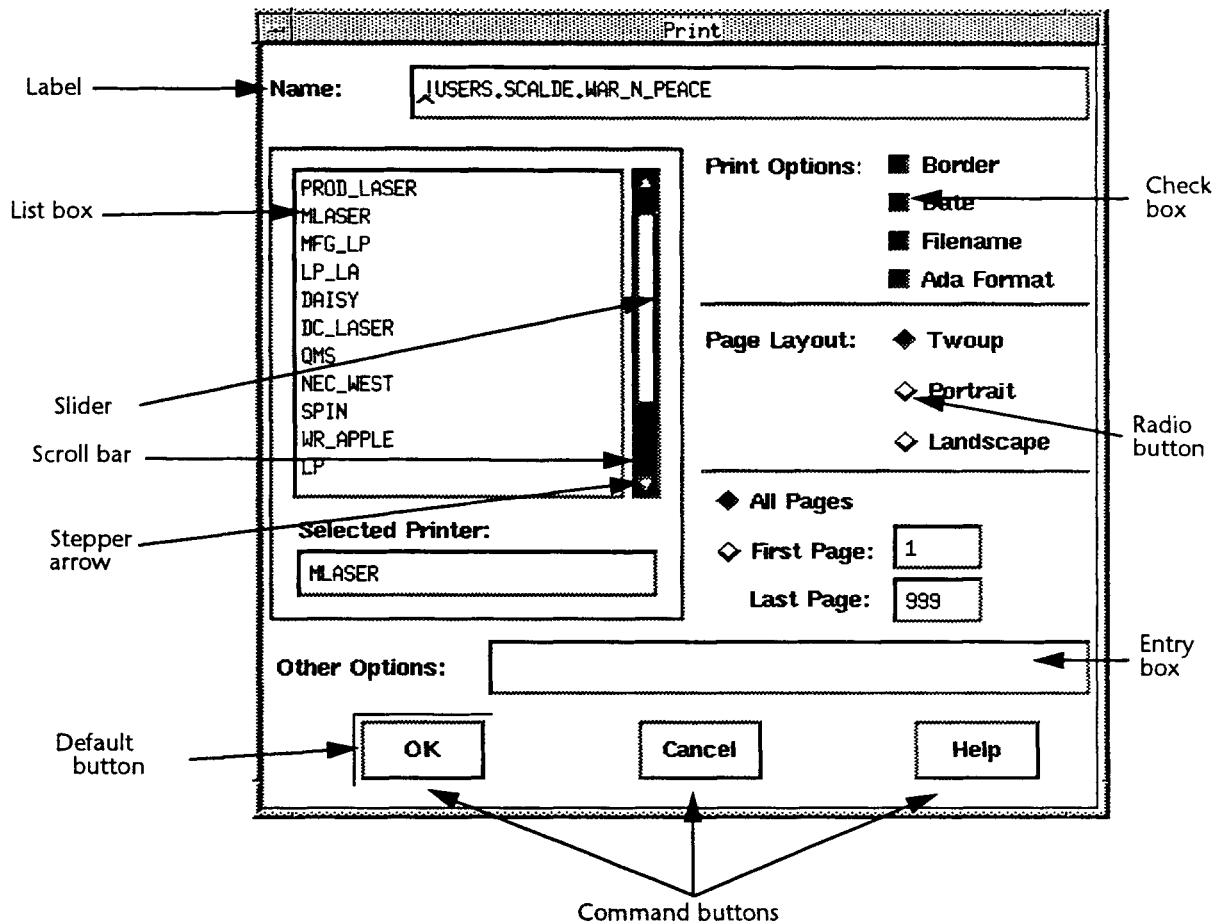


Figure B-2 A Typical Access Dialog Box

- *Labels* represent the name of a particular part of a dialog box. In the Print dialog box example, the label refers to the Name entry box.
- *List boxes* present a scrollable list of items from which to choose. Typically the chosen item appears in a related text-entry box.
- *Scroll bars* are used to change a user's viewpoint of a list. You can click the *scroll arrows* to move the *slider*, or drag the slider itself.
- *Command buttons* are controls that cause immediate action. The following command buttons appear at the bottom of most dialog boxes:
 - OK, which causes the chosen command to execute
 - Cancel, which closes the dialog box and terminates the chosen command
 - Help, which displays information about the dialog box in the Access help window
- The *default button* is highlighted and can be activated by pressing [Return], regardless of current keyboard focus within the dialog box. The only exception is when you are entering text in an entry box (see below) that accepts [Return] as a literal text entry (for example, the Comments entry box in CMVC:Check In).

- *Entry boxes* are where you type information such as filenames.
 - The information inside a text-entry box is the text field displayed by the box. If the text field is longer than the box itself, you can scroll it.
 - An insertion point (^) indicates where characters are inserted in the text.
- *Radio buttons* set options that affect a command's behavior. Within a group of radio buttons, only one can be selected (indicated by the solid diamond).
- *Check boxes* also set options that affect a command's behavior; however, unlike radio buttons, any number of check boxes can be selected (indicated by a solid square).
- *Scale bars* (not pictured) provide numeric input to a command. The number specified by a scale bar is determined by the position of the slider inside the bar.
- *Option menus* (not pictured) reveal a list of commands or other selections. To display an option menu, click the *option button* (distinguished by a graphic box next to the label). The new selection becomes the label on the option button.
- *Popup menus* (not pictured) are menus inside a dialog box. Click on the menu to reveal the menu commands, and then click on the command to execute it. Popup menus have an arrow symbol (=>) next to the label.

Using the Mouse to Respond to Dialog Boxes

Text-Entry Boxes

- To start typing in a text-entry box, put the pointer in the text-entry box and click; start typing.
- To erase adjacent characters in a text-entry box, put the pointer on the first unwanted character, press the left mouse button, and drag the pointer across the rest of the unwanted characters. The selected characters disappear when you start to type on them.
- To erase the entire contents of a text-entry box, put the pointer in the text-entry box and double-click. The contents of the box appear in reverse video to indicate that they have been selected using Motif selection. The characters in the Motif selection disappear when you start to type.

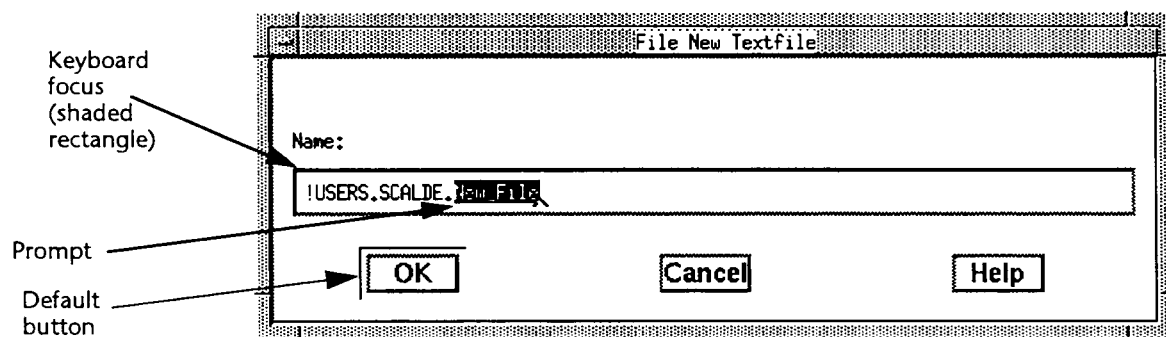



Figure B-3 A Typical Dialog Box with a Text-Entry Box, Command Buttons, and a Motif-Selected Prompt

Other Controls

- To initiate the action of a command button, put the pointer on the button and click.
- To specify an option associated with a radio button, put the pointer in the empty diamond and click. The diamond is filled in with a solid color and the previously selected radio button in the same group of buttons is deselected (its solid diamond becomes an outline).
- To specify an option associated with a check box, put the pointer in the empty box and click. The box is filled in with a solid color.
- To select an item from a list box, put the pointer on the desired item and click. The item is displayed in reverse video in the list and appears in the associated text-entry box.
- To deselect an item from a list box, put the pointer on the selected item and double-click.
- To specify a number using a scale bar, put the pointer on the slider and drag it in the appropriate direction until the desired number appears beside the scale bar.
- To activate a window-control button or user-defined button, click the button.
- To activate a search arrow in a dialog box, click the arrow to search for the next specified item in the direction the arrow is pointing.
- To remove a help window or a control palette, click: 

Using the Keyboard to Respond to Dialog Boxes

You can use keyboard alternatives to navigate a dialog box, specify information, and initiate the actions of command buttons, all without using a mouse. (If you have customized key bindings, keep in mind the keys you would actually press.)

Note: *The operations in this section do not apply if you have set your keyboard-focus policy to pointer; you must first explicitly move the keyboard focus to the particular dialog box. Once the keyboard focus is on the box, you can execute these operations. Contact your UNIX system administrator for help on focus policy.*

Navigating a Dialog Box

When you use keyboard shortcuts to respond to a dialog box, you must establish which element of the dialog box has the *keyboard focus*—that is, which element is to receive input from the keyboard. The keyboard focus is indicated by a shaded rectangle around the element (see Figure B-2).

When a dialog box first appears, the keyboard focus is usually around the OK button. You can move the keyboard focus to other elements and groups of elements in the dialog box. To do this, press [Tab] to move the focus forward around the dialog box, and press [Shift][Tab] to move the focus in the reverse direction.

For example, choosing the File:New:Text File command brings up a dialog box (see Figure B-2) in which the keyboard focus is on the text-entry box. Start typing, and the prompt disappears. Press [Tab] to move the keyboard focus to the group of command buttons.

Note that [Tab] and [Shift][Tab] alone do not move to every element in the box. Rather, these keys move the focus among:

- Individual text-entry boxes and list boxes
- Groups of radio buttons, check boxes, or command buttons. Each group is called a *field*. (Within each group, a predetermined element receives the keyboard focus.)

To move the keyboard focus among the elements within a group or among the items within a list box, press the arrow keys to move the focus up, down, left, or right, as appropriate.

For example, use [→] to move the focus from the OK button to the Cancel button within the group of command buttons at the bottom of a dialog box.

Note: You can also use [Control][Tab] instead of [Tab] to navigate in a dialog box. This is useful because some entry boxes (for example, the Comments entry box in the CMVC:Check In and Check Out dialog boxes) accept [Tab] as a literal part of the text entry and do not move the location cursor to the next field.

Specifying Information

To toggle the settings of check boxes and radio buttons, or to select an item from a list box:

To move the keyboard focus to the desired box, button, or list box item, use [Tab] or [Control][Tab] and/or the arrow keys.

To select or deselect the element, press [Space Bar] (this is equivalent to putting the mouse pointer on it and clicking).

Initiating the Action of a Command Button

Within a dialog box, one command button is designated as the *default button*. As shown in Figure B-2, the default button has a special outline that is different from the keyboard-focus border. A given dialog box has only one default button.

To initiate the action of a default button, press [Return]. You do not need to move the keyboard focus to the default button first. This is equivalent to putting the pointer on the button and clicking.

For example, when you finish typing in the text-entry box of the File:New:Text File command's dialog box, you can press [Return] to activate the default button (OK) while the keyboard focus remains on the text-entry box.

To initiate the action of another command button, you first turn it into the default button:

1. Use [Tab] and/or the arrow keys to move the keyboard focus to the desired command button. Moving the keyboard focus to a command button automatically moves the default designation as well.
2. Press [Return].

For example, if you wanted to cancel the File:New:Text File command's dialog box instead of activating it, you would move the keyboard focus to the Cancel button and press [Return].

Shortcut for Canceling a Dialog Box

Note that pressing [Escape] cancels a dialog box regardless of the keyboard focus or default designation.

Table B-4 Summary of Keys for Responding to Dialog Boxes

Note: If you have customized your key bindings, keep in mind the keys you would actually press

Key	Action
[Tab] or [Control][Tab]	Moves the keyboard focus forward to an element or group of elements in the dialog box
[Shift][Tab]	Moves the keyboard focus back to an element or group of elements in the dialog box
[←] [→] [↑] [↓]	Moves the keyboard focus among elements in a group
[Space Bar]	Selects the element with the keyboard focus
[Return]	Initiates the action of the default button
[Escape]	Cancel the dialog box

Table B-5 Summary of Keys for Editing Text-Entry Boxes

Note: If you have customized your key bindings, keep in mind the keys you would actually press.

Key	Action
[←] [→]	Moves the insertion point in the direction of the arrow, scrolling if necessary
[Home]	Moves the insertion point to the beginning of the field
[End]	Moves the insertion point to the end of the field
[Shift][←] [Shift][→]	Selects characters in the direction of the arrow, starting at the insertion point; typing deletes the selected characters
[Shift][Home]	Selects characters from the insertion point to the beginning of the field; typing deletes the selected characters
[Shift][End]	Selects characters from the insertion point to the end of the field; typing deletes the selected characters
[Delete] or [Back Space]	Deletes previous character
[Control][D]	Deletes next character

C

Access Equivalents: Environment Commands

This appendix lists basic Environment commands and the Access menu items and buttons that are most closely related to them. This appendix will help if you know the Environment command that you would enter in a command window and are trying to perform the same operation through the Access menus and buttons.

Note that many of these operations can also be performed through key and mouse bindings. If an operation is available *only* through key and mouse bindings, that is noted. (See Appendix D or the Rational Access Quick Reference for key and mouse bindings.) Environment commands that are not listed in this appendix must be executed through command windows.

If you want to start with the name of an Access menu item and find out which Environment command or commands are similar to that menu item, see the Access online help.

Table C-1 Environment Commands and Access Equivalents

Environment Command	Access Menu Items or Buttons
Abbreviations.Print	File:Print
Access_List.Set	File:Properties
Access_List.Set_Default	File:Properties
Activity.Create	File:New:Activity
Activity.Edit	CMVC:Activity
Activity.Set_Default	Session:Profile
Ada.Create_Body	Program:Build:Build Body
Ada.Create_Private	Program:Build:Build Private Part
Ada.Make_Inline	Program:Build:Make Inline
Ada.Make_Separate	Program:Build:Make Separate
Ada.Other_Part	Navigate:Other Part
Ada.Show_Unused	Program:Show Unused
Ada.Show_Usage	Program:Show Usage
Ada.Withdraw	Program:Build:Withdraw
Command.Debug	Debug:Start Debugging of Command
Common.Abandon	File:Close
Common.Clear_Underlining	Edit:Underlines Off
Common.Commit	File:Save
Common.Complete	Program:Complete
Common.Create_Command	Not available from menus or buttons. See Function Key Palette online.

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Common.Definition In_Place => False In_Place => True	File:Browse Navigate:Definition Not available from menus or buttons. See Function Key Palette online.
Common.Demote	Program:Demote
Common.Edit Open object for editing Clear command window Withdraw Ada construct for editing Edit entry in an activity Edit entry in a searchlist Edit entry in a switch file	File:Open File:Revert Program:Incremental:Incremental Edit Edit:Edit Entry Edit:Edit Entry Edit:Edit Entry
Common.Elide	Not available from menus or buttons. See object operations in Appendix D.
Common.Enclosing In_Place => False In_Place => True	Navigate:Enclosing Not available from menus or buttons. See Function Key Palette online.
Common.Expand	Not available from menus or buttons. See object operations in Appendix D.
Common.Explain	Help:Explain
Common.Format Format Ada unit Format library or mailbox	Program:Format File:Revert
Common.Insert_File	File:Insert File
Common.Object.Child	Not available from menus or buttons. See object operations in Appendix D.
Common.Object.Copy	File:Copy File
Common.Object.Delete Delete any object Delete construct from Ada unit Delete mail message Delete entry from an activity Delete entry from a set of links Delete entry from a searchlist Delete entry from a switch file	File>Delete File Program:Incremental:Incremental Delete Tools:Mail>Delete Mail Message Edit>Delete Entry Edit>Delete Entry Edit>Delete Entry Edit>Delete Entry
Common.Object.First_Child	Not available from menus or buttons. See object operations in Appendix D.
Common.Object.Insert Create new Ada unit Add construct to Ada unit Add entry to an activity Add entry to a set of links Add entry to a searchlist Add entry to a switch file	File:New:Ada Program:Incremental:Incremental Insert Edit:Add Entry Edit:Add Entry Edit:Add Entry Edit:Add Entry
Common.Object.Last_Child	Not available from menus or buttons. See object operations in Appendix D.
Common.Object.Move	File:Move File

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Common.Object.Next	Not available from menus or buttons. See object operations in Appendix D.
Common.Object.Parent	Not available from menus or buttons. See object operations in Appendix D.
Common.Object.Previous	Not available from menus or buttons. See object operations in Appendix D.
Common.Promote Close any object Promote Ada unit Promote Ada unit stub Send mail message	File:Close Program:Promote Program:Incremental:Incremental Promote Tools:Mail:Send Mail Message
Common.Redo	Not available from menus or buttons. See object operations in Appendix D.
Common.Release	File:Close
Common.Revert	File:Revert
Common.Semanticize Check most objects Check mail message	Program:Semanticize Tools:Mail:Check Mail Message
Common.Sort_Image Sort mailbox Sort other objects	Tools:Mail:Sort Mailbox Not available from menus or buttons. See object operations in Appendix D.
Common.Undo Undelete mail message Undo other operation	Tools:Mail:Undelete Mail Message Not available from menus or buttons. See object operations in Appendix D.
Compilation.Demote Goal => Installed Goal => Source Goal => Archived	Program:Demote to Installed Program:Demote to Source Program:Demote to Archived
Compilation.Destroy	File>Delete File
Compilation.Load	Program:Build:Load
Compilation.Parse	Program:Build:Parse Source Files
Compilation.Promote Goal => Source Goal => Installed Goal => Coded	Program:Promote to Source Program:Promote to Installed Program:Promote to Coded
Cmvc.Abandon_Reservation	CMVC:Abandon
Cmvc.Accept_Changes	CMVC:Accept Changes
Cmvc.Check_In	CMVC:Check In File:Properties
Cmvc.Check_Out	CMVC:Check Out File:Open File:Properties

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Cmvc.Copy Create_Combined_View => True Create_Load_View => True Create_Spec_View => True Any kind and freeze copy	File:New:Working View File:New:Working View File:New:Spec View File:New:Release View
Cmvc.Destroy_Subsystem	File>Delete File
Cmvc.Destroy_System	File>Delete File
Cmvc.Destroy_Views	File>Delete File
Cmvc.Edit	CMVC:CMVC Editor
Cmvc.Import	CMVC:Imports/Model
Cmvc.Information Show_Exported_Units => True Show_Imports => True Show_Model => True Show_Referencers => True	CMVC:CMVC Report:List View Exports CMVC:CMVC Report:List View Imports CMVC:CMVC Report:List View Model CMVC:CMVC Report:List View Referencers
Cmvc.Initial System_Object_Type => Cmvc.Combined_Subsystem Cmvc.Spec_Load_Subsystem Cmvc.System	File:New:Subsystem File:New:Subsystem File:New:System
Cmvc.Join	CMVC:Join
Cmvc.Make_Code_View	File:New:Code View
Cmvc.Make_Controlled	CMVC:Control/Uncontrol File:Properties
Cmvc.Make_Path	File:New:Working View
Cmvc.Make_Spec_View	File:New:Spec View
Cmvc.Make_Subpath	File:New:Working View
Cmvc.Make_Uncontrolled	CMVC:Control/Uncontrol File:Properties File>Delete File
Cmvc.Release	File:New:Release View
Cmvc.Revert	CMVC:Revert to Generation
Cmvc.Sever	CMVC:Sever
Cmvc.Show_All_Checked_Out	CMVC:CMVC Report:List Checked-Out Objects
Cmvc.Show_All_Controlled	CMVC:CMVC Report:List Controlled
Cmvc.Show_All_Uncontrolled	CMVC:CMVC Report:List Uncontrolled
Cmvc.Show_History	CMVC:CMVC Report:Object History Information
Cmvc.Show_Out_Of_Date_Objects	CMVC:CMVC Report:List Out-of-Date Objects
Debug.Activate	Activate Breaks and Activate All Breaks buttons in the Breakpoints buttons on the Debugger Palette
Debug.Address_To_Location	Address To Location... button in the Machine and Memory buttons on the Debugger Palette
Debug.Attach_Process	Attach Process button in the Target buttons on the Debugger Palette

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Debug.Break	Break..., Break Here, and Temporary Break Here buttons in the Breakpoints buttons on the Debugger Palette
Debug.Catch	Catch... and Catch Unlisted buttons in the Exception Handling buttons on the Debugger Palette
Debug.Clear_Stepping	Clear Stepping button in the Debugger Control buttons on the Debugger Palette
Debug.Connect	Connect button in the Target buttons on the Debugger Palette
Debug.Context	Set Control Context and Set Evaluation Context buttons in the Debugger Control buttons on the Debugger Palette
Debug.Current_Debugger	Debug:Debugger Window
Debug.Debug_Off Kill_Jobs => False Kill_Jobs => True	Debug:Finish Debugging Job and Detach Detach Job button in the Quit buttons on the Debugger Palette Debug:Finish Debugging Job and Kill Kill Job Being Debugged button in the Quit buttons on the Debugger Palette
Debug.Disable	Disable... button in the Debugger Control buttons on the Debugger Palette
Debug.Enable	Enable... button in the Debugger Control buttons on the Debugger Palette
Debug.Exception_To_Name	Exception To Name... button in the Machine and Memory buttons on the Debugger Palette
Debug.Execute	Continue and Continue All buttons on the Debugger Palette and the Continue Task button in the Task Control buttons on the Debugger Palette
Debug.Forget	Forget... and Forget All buttons in the Exception Handling buttons on the Debugger Palette
Debug.Hold	Hold Task and Hold All buttons in the Task Control buttons on the Debugger Palette
Debug.Information Info_Type => Debug.Exceptions Info_Type => Debug.Rendezvous Info_Type => Debug.Space	Buttons in the Task Information buttons on the Debugger Palette: Exceptions - All Tasks Rendezvous Info - All Tasks Space Information - All Tasks
Debug.Invoke	Invoke... button in the Target buttons on the Debugger Palette
Debug.Kill	Terminate Debugger button in the Quit buttons on the Debugger Palette
Debug.Location_To_Address	Location To Address... and Object To Address... buttons in the Machine and Memory buttons on the Debugger Palette
Debug.Memory_Display	Display Memory... button in the Machine and Memory buttons on the Debugger Palette

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Debug.Memory_Modify	Modify Memory... button in the Machine and Memory buttons on the Debugger Palette
Debug.Modify	Modify... button on the Debugger Palette
Debug.Propagate	Propagate... and Propagate Unlisted buttons in the Exception Handling buttons on the Debugger Palette
Debug.Put	Put and Put... buttons on the Debugger Palette
Debug.Reconnect	Reconnect button in the Target buttons on the Debugger Palette
Debug.Register_Display	Display Registers... button in the Machine and Memory buttons on the Debugger Palette
Debug.Register_Modify	Modify Register... button in the Machine and Memory buttons on the Debugger Palette
Debug.Release	Release Task and Release All buttons in the Task Control buttons on the Debugger Palette
Debug.Remove Delete => False Delete => False, Breakpoint => 0 Delete => True Delete => True, Breakpoint => 0	Buttons in the Breakpoints buttons on the Debugger Palette: Deactivate Break Deactivate All Breaks Delete Break Delete All Breaks
Debug.Reset_Defaults	Reset to Defaults button in the Debugger Control buttons on the Debugger Palette
Debug.Run Stop_At => Debug.Local_Statement Stop_At => Debug.Returned Stop_At => Debug.Statement	Buttons on the Debugger Palette: Step Local Step Ret Step Stmt
Debug.Set_Value	Set... button in the Debugger Control buttons on the Debugger Palette
Debug.Show Values_For => Debug.All_State Values_For => Debug.Breakpoints Values_For => Debug.Contexts Values_For => Debug.Exceptions Values_For => Debug.Flags Values_For => Debug.Libraries Values_For => Debug.Stops_And_Holds Values_For => Debug.Traces	Menu items and buttons on the Debugger Palette: Show:All Debugger State Show Breakpoints button in the Breakpoints buttons Show Contexts button in the Debugger Control buttons Show Exceptions button in the Exception Handling buttons Show Debugger Switches button in the Debugger Control buttons Show:Libraries Show Stops and Holds button in the Task Control buttons Show Tracing button in the Tracing buttons
Debug.Source	Source button on the Debugger Palette
Debug.Stack	Stack button on the Debugger Palette

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Debug.Stop Current task Specific task All tasks	Buttons on the Debugger Palette: Stop Stop Task in the Task Control buttons Stop All
Debug.Target_Request	Target Request... button in the Target buttons on the Debugger Palette
Debug.Task_Display Task_Set => Debug.All_Tasks Task_Set => Debug.Blocked Task_Set => Debug.Held Task_Set => Debug.Not_Running Task_Set => Debug.Running Task_Set => Debug.Stopped	Menu items and buttons on the Debugger Palette: Show:All Tasks List All Tasks button in the Task Information buttons List Blocked Tasks button in the Task Information buttons Show:Held Tasks List Held Tasks button in the Task Information buttons List Tasks Not Running button in the Task Information buttons List Running Tasks button in the Task Information buttons Show:Stopped Tasks List Stopped Tasks button in the Task Information buttons
Debug.Trace Any parameter settings On => True On => False	Buttons in the Tracing buttons on the Debugger Palette: Tracing Control Activate All Tracing Deactivate All Tracing
Debug.Trace_To_File	Tracing Control button in the Tracing buttons on the Debugger Palette (choose from the options menu)
Dependents.Show	Program:Show Usage
Editor.Char.Quote	Not available from menus or buttons. See Quick Reference for key binding.
Editor.Char.Tab_To_Comment	Not available from menus or buttons. See object operations in Appendix D.
Editor.Char.Transpose	Not available from menus or buttons. See Quick Reference for key binding.
Editor.Cursor.Next Prompt => True, Underline => True Prompt => False, Underline => True	Navigate:Next Item Navigate:Next Underline
Editor.Cursor.Previous Prompt => True, Underline => True Prompt => False, Underline => True	Navigate:Previous Item Navigate:Previous Underline
Editor.Hold_Stack.Copy_Top	Not available from menus or buttons. See region operations in Appendix D.
Editor.Hold_Stack.Delete_Top	Not available from menus or buttons. See region operations in Appendix D.
Editor.Hold_Stack.Next	Not available from menus or buttons. See region operations in Appendix D.

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Editor.Hold_Stack.Previous	Not available from menus or buttons. See region operations in Appendix D.
Editor.Hold_Stack.Push	Edit:Copy
Editor.Hold_Stack.Rotate	Not available from menus or buttons. See region operations in Appendix D.
Editor.Hold_Stack.Swap	Not available from menus or buttons. See region operations in Appendix D.
Editor.Hold_Stack.Top	Edit:Paste
Editor.Image.Beginning_Of	Top of Image button
Editor.Image.Down	Scroll Down button
Editor.Image.End_Of	Bottom of Image button
Editor.Image.Left	Scroll Left button
Editor.Image.Right	Scroll Right button
Editor.Image.Up	Scroll Up button
Editor.Key.Name	Help:On Key
Editor.Key.Prompt	Not available from menus or buttons. See Function Key Palette online.
Editor.Line.Beginning_Of	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Capitalize	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Center	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Copy	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Delete	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Delete_Forward	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.End_Of	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Insert	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Join	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Lower_Case	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Open	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Transpose	Not available from menus or buttons. See line operations in Appendix D.
Editor.Line.Upper_Case	Not available from menus or buttons. See line operations in Appendix D.
Editor.Macro.Bind	Tools:Macro:Bind Macro to Key
Editor.Macro.Execute	Tools:Macro:Execute Macro

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Editor.Macro.Finish	Tools:Macro:End Macro Def
Editor.Macro.Start	Tools:Macro:Begin Macro Def
Editor.Mark.Copy_Top	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Delete_Top	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Next	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Previous	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Push	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Rotate	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Swap	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Mark.Top	Not available from menus or buttons. See mark operations in Appendix D.
Editor.Quit	File:Exit
Editor.Region.Beginning_Of	Navigate:Top of Region
Editor.Region.Capitalize	Edit:Capitalize
Editor.Region.Comment	Edit:Make into Comment
Editor.Region.Copy	Edit:Copy and Edit:Paste in combination
Editor.Region.Delete	Edit:Cut
Editor.Region.End_Of	Navigate:Bottom of Region
Editor.Region.Fill	Edit:Fill
Editor.Region.Finish	Not available from menus or buttons. See region operations in Appendix D.
Editor.Region.Justify	Edit:Justify
Editor.Region.Lower_Case	Edit:Lowercase
Editor.Region.Move	Edit:Cut and Edit:Paste in combination
Editor.Region.Off	Edit:Underlines Off
Editor.Region.Start	Not available from menus or buttons. See region operations in Appendix D.
Editor.Region.Uncomment	Edit:Uncomment
Editor.Region.Upper_Case	Edit:Uppercase
Editor.Screen.Clear	Session:Screen:Full Reset
Editor.Screen.Push	Session:Screen:Screen Push
Editor.Screen.Redraw	Not available from menus or buttons. See Quick Reference for key binding.
Editor.Screen.Top	Session:Screen:Screen Pop
Editor.Search.Next	Edit:Search/Replace
Editor.Search.Previous	Edit:Search/Replace
Editor.Search.Replace_Next	Edit:Search/Replace

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Editor.Search.Replace_Previous	Edit:Search/Replace
Editor.Set.Designation_Off	Edit:Deselect Edit:Underlines Off
Editor.Set.Fill_Column	Edit:Typing Modes
Editor.Set.Fill_Mode	Edit:Typing Modes
Editor.Set.Insert_Mode	Edit:Typing Modes
Editor.Set.Window_Frames	Session:Screen:Window Frames
Editor.Window.Beginning_Of	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Child	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Copy	Copy Window button
Editor.Window.Delete	Remove Window button
Editor.Window.Demote	Unlock Window button
Editor.Window.Directory	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.End_Of	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Expand Lines => 4 Lines => -4 Lines => 99	Expand Window button Shrink Window button Fully Expand Window button
Editor.Window.Focus	Realign Windows button
Editor.Window.Join Repeat => 1 Repeat => -1	Join Next Window button Join Previous Window button
Editor.Window.Next	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Parent	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Previous	Not available from menus or buttons. See window operations in Appendix D.
Editor.Window.Promote	Lock Window button
Editor.Window.Transpose	Not available from menus or buttons. See window operations in Appendix D.
Editor.Word.Beginning_Of	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Capitalize	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Copy	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Delete	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Delete_Backward	Not available from menus or buttons. See Quick Reference for key binding.

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Editor.Word.Delete_Forward	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.End_Of	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Lower_Case	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Next	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Previous	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Transpose	Not available from menus or buttons. See word operations in Appendix D.
Editor.Word.Upper_Case	Not available from menus or buttons. See word operations in Appendix D.
Job.Connect	Session:Jobs:Connect to Job
Job.Disable	Session:Jobs:Disalbe Job
Job.Disconnect	Session:Jobs:Disconnect Current Job
Job.Enable	Session:Jobs:Enable Job
Job.Interrupt	Session:Jobs:Disconnect Current Job
Job.Kill	Session:Jobs:Kill Job
Library.Copy	File:Copy File
Library.Create_Directory	File:New:Directory
Library.Create_World	File:New:World
Library.Destroy	File>Delete File
Library.Freeze	File:Properties
Library.Move	File:Move File
Library.Resolve	Navigate:Resolve Name
Library.Unfreeze	File:Properties
Log.Set_Output (for compilation)	Program:Promote (Log File option)
Mail.Answer To_All => False To_All => True	Tools:Mail:Reply Tools:Mail:Reply to All
Mail.Create Main for new user Other for existing user	Tools:Operator:Create/Edit User File:New:Mailbox
Mail.Edit	Tools:Mail:Read Mail
Mail.Expunge	Tools:Mail:Expunge Mailbox
Mail.Forward	Tools:Mail:Forward
Mail.Remail	Tools:Mail:Remail
Mail.Reply To_All => False To_All => True	Tools:Mail:Reply Tools:Mail:Reply to All
Mail.Send	Tools:Mail:New Mail Message

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Operator.Add_To_Group add single or multiple users add single user	Tools:Operator:Create/Edit Group Tools:Operator:Create/Edit User
Operator.Cancel_Shutdown	Tools:Operator:Cancel Shutdown
Operator.Change_Password	Tools:Operator:Create/Edit User
Operator.Create_Group	Tools:Operator:Create/Edit Group
Operator.Create_User	Tools:Operator:Create/Edit User
Operator.Delete_Group	Tools:Operator:Create/Edit Group
Operator.Delete_User	Tools:Operator:Create/Edit User
Operator.Disk_Space	Tools:System Info:Disk Tools:System Info:Configuration
Operator.Force_Logoff	Tools:Operator:Force Logoff
Operator.Remove_From_Group Remove single or multiple users Remove single user	Tools:Operator:Create/Edit Group Tools:Operator:Create/Edit User
Operator.Show_Shutdown_Settings	Tools:Operator:Shutdown System
Operator.Shutdown	Tools:Operator:Shutdown System
Operator.Shutdown_Warning	Tools:Operator:Shutdown System
Profile.Get_Default	Session:Profile
Profile.Include_In_Default	Session:Profile
Profile.Set_Default	Session:Profile
Profile.Set_Default_Activity	Session:Profile
Profile.Set_Default_Filter	Session:Profile
Profile.Set_Default_Log_File	Session:Profile
Profile.Set_Default_Prefixes	Session:Profile
Profile.Set_Default_Reaction	Session:Profile
Profile.Set_Default_Remote_Passwords	Session:Profile
Profile.Set_Default_Remote_Sessions	Session:Profile
Profile.Set_Default_Response	Session:Profile
Profile.Set_Default_Width	Session:Profile
Program.Create_Job	File:Run
Program.Run	File:Run
Queue.Print	File:Print
Remote.Run	File:Run (Alternate Machine option)
Remote_Passwords.Set_Default	Session:Profile
Search_List.Show_List	Session:Searchlist
Speller.Check_Image	Edit:Spelling:Check Image
Speller.Check_Text	Edit:Spelling:Check Word
Speller.Exchange_Word	Edit:Spelling:Replace Word
Speller.Explain_Next	Edit:Spelling:Next Spelling Error
Speller.Learn_Replacement	Edit:Spelling:Learn Replacement
Speller.Learn_Word	Edit:Spelling:Add Word to Dict

Table C-1 Environment Commands and Access Equivalents (continued)

Environment Command	Access Menu Items or Buttons
Speller.Speller_Window	Not available from menus or buttons. See word operations in Appendix D.
Switches.Associate	File:Properties
Switches.Create	File:New:Switch File
Switches.Edit	Program:Compiler Switches
Switches.Edit_Session_Attributes Any session switch A Profile.@ switch	Session:Session Switches Session:Profile
System_Availability'Spec_View.Units.System- _Report.Generate	Tools:Operator:Report Generation
System_Backup.Backup	Tools:Operator:Backup
System_Backup.History	Tools:Operator:Backup History
System_Maintenance'Spec_View.Units- .Verify_Backup	Tools:Operator:Verify Backup
System_Uilities.System_Boot_Configuration	Tools:System Info:Configuration
System_Uilities.Terminal	Tools:System Info:Users
Text.Create	File:New:Text File
Text.End_Of_Input	Session:End-of-Input
What.Command	Help:On Environment
What.Does	Help:On Environment
What.Home_Library	Navigate:Home Library
What.Jobs My_Jobs_Only => True My_Jobs_Only => False	Session:Jobs:Show Jobs Session:Jobs:Show All Jobs
What.Line	Not available from menus or buttons. See line operations in Appendix D.
What.Load	Tools:System Info:System Load
What.Locks	Tools:Files Info:Locks
What.Object	Tools:Files Info:Name/Version
What.Tabs	Not available from menus or buttons. See Quick Reference for key binding.
What.Users	Tools:System Info:Users
What.Version	Tools:System Info:Configuration
Work_Order.Create	File:New:Work Order
Work_Order.Create_List	File:New:Work Order List
Work_Order.Create_Venture	File:New:Venture

D

Access Equivalents: Key Bindings

Different keyboards have different configurations of physical keys. For this reason, Rational's training and documentation refer to *logical keys*. A logical key names an operation in the Environment that is generally bound to one or more physical keys. The physical keys to which a particular logical key is bound depends on the interface you are using. For example, [Definition] is a logical key that displays the object indicated by the cursor; in Access, [Definition] is bound to the physical key [F5].

This appendix lists the logical key names used in Rational's training and documentation and the Access key bindings, mouse buttons, menu items, and menu buttons that are most closely related to them. Note that unlike the Rational X Interface (RXI) and Rational Windows Interface (RWT), which are delivered with several sets of key bindings to support keyboard variants, Access is delivered with a single set of key bindings. These bindings use only the function keys, arrow keys, and alphanumeric keys with the [Shift], [Control], and [Meta] modifiers.

This appendix contains the following tables:

- Table D-1, "Fundamental Set of Logical Keys," on page 152
- Table D-2, "Object Operations," on page 155
- Table D-3, "Region Operations," on page 157
- Table D-4, "Window Operations," on page 160
- Table D-5, "Image Operations," on page 163
- Table D-6, "Line Operations," on page 165
- Table D-7, "Word Operations," on page 167
- Table D-8, "Mark Operations," on page 169

For a quick look at function key bindings, including the item operations available, you can display the Function Key Palette from the main Access window. For a quick look at all the operations bound to Access keyboard and mouse operations, see the Rational Access Quick Reference.

FUNDAMENTAL SET OF LOGICAL KEYS

Table D-1 Fundamental Set of Logical Keys

Logical Key	Physical Key or Mouse Button	Menu Item or Button
[Begin Of]	[Home]	Not applicable
[Break]	Not available	Break Here button in the Breakpoints buttons on the Debugger Palette
[Code (This World)]	Not available	Program:Promote to Coded
[Code Unit]	Not available	Program:Promote to Coded
[Commit]	[Control][Return]	File:Save
[Complete]	[Shift][F6]	Program:Complete
[Create Ada]	[Shift][F11]	File:New:Ada
[Create Body]	Not available	Program:Build:Build Body
[Create Command]	[F6]	Not available
[Create Directory]	Not available	File:New:Directory
[Create Text]	[F11]	File:New:Text File
[Create World]	Not available	File:New:World
[Debug]	[Meta][Return]	Debug:Start Debugging of Command
[Definition]	[F5] dbl click left	Navigate:Definition
[Definition (In Place)]	[Shift][F5] [Shift]+dbl click left	Not available
[Demote]	[Shift][F8]	Program:Demote
[Edit]	[Shift][F2]	Program:Incremental:Incremental Edit
[Enclosing]	[F7] dbl click right	Navigate:Enclosing
[Enclosing (In Place)]	[Shift][F7] [Shift]+dbl click right	Not available
[End Of]	[End]	Not applicable
[Explain]	[F3]	Help:Explain
[Format]	[F9]	Program:Format
[Go]	Not available	Continue button on the Debugger Palette
[Help on Command]	[Help]	Help:On Environment
[Help on Key]	[Control][Q]	Help:On Key

Table D-1 Fundamental Set of Logical Keys (continued)

Logical Key	Physical Key or Mouse Button	Menu Item or Button
[Help Window]	Not available	Not available
[Home Library]	[Shift][F10]	Navigate:Home Library
[Image]	[Control][F4]	Most [Image] operations can be found on the window-control button panel
[Install Unit]	Not available	Program:Promote to Installed Program:Demote to Installed
[Item Off]	[Control][X]	Edit:Deselect
[Line]	[Control][F5]	Most [Line] operations can be performed by using [Control][Meta] in combination with an alphanumeric operation key
[Mark]	[Control][F7]	Macro operations are available on the Tools:Macro menu; mark operations are not available
[Next Item]	[F12]	Navigate:Next Item
[Object]	[Control][F1]	Most [Object] operations can be performed from the File menu
[Other Part]	Not available	Navigate:Other Part
[Previous Item]	[Shift][F12]	Navigate:Previous Item
[Promote]	[F8]	Program:Promote
[Prompt For]	[Shift][F1]	Not available
[Put]	Not available	Put button on the Debugger Palette
[Replace]	[Control][Meta][S]	Edit:Search/Replace
[Replace (Backward)]	[Control][Meta][R]	Edit:Search/Replace
[Region]	[Control][F2]	Most [Region] operations can be performed from the Edit menu
[Search]	[Control][S]	Edit:Search/Replace
[Search (Backward)]	[Control][R]	Edit:Search/Replace
[Semanticize]	[Shift][F9]	Program:Semanticize
[Show Source]	Not available	Source button on the Debugger Palette
[Source (This World)]	Not available	Program:Promote to Source Program:Demote to Source
[Source Unit]	Not available	Program:Promote to Source Program:Demote to Source
[Stack]	Not available	Stack button on the Debugger Palette

Table D-1 Fundamental Set of Logical Keys (continued)

Logical Key	Physical Key or Mouse Button	Menu Item or Button
[Step]	Not available	Step Stmt button on the Debugger Palette
[Step Local]	Not available	Step Local button on the Debugger Palette
[Window]	[Control][F4]	Most [Window] operations can be performed from the window-control button panel
[Word]	[Control][F6]	Most [Word] operations can be performed using [Meta] in combination with an alphanumeric operation key

OBJECT OPERATIONS

Table D-2 Object Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Object] - [A]	Select first child	[Control][F1] - [A] [Control][F1] - [B]	Not available
[Object] - [B]	Select first child	[Control][F1] - [A] [Control][F1] - [B]	Not available
[Object] - [C]	Copy object	[Control][F1] - [C]	File:Copy File
[Object] - [D]	Delete object	[Control][F1] - [D] [Control][F1] - [K]	File>Delete File Edit>Delete Entry Tools:Mail>Delete Mail Message Program:Incremental:Incremental Delete
[Object] - [E]	Select last child	[Control][F1] - [E]	Not available
[Object] - [G]	Abandon object	[Control][F1] - [G]	File:Close
[Object] - [H]	Select parent	[Control][←] [Control][F1] - [←] [Control][F1] - [H] [Control]+dbl click left	Not available
[Object] - [I]	Insert object	[Shift][F11] [Control][F1] - [I]	File:New:Ada Edit:Add Entry Program:Incremental:Incremental Insert
[Object] - [J]	Select child	[Control][→] [Control][F1] - [→] [Control][F1] - [J] [Control]+dbl click right	Not available
[Object] - [K]	Delete object	[Control][F1] - [D] [Control][F1] - [K]	File>Delete File Edit>Delete Entry Tools:Mail>Delete Mail Message Program:Incremental:Incremental Delete
[Object] - [M]	Move object	[Control][F1] - [M]	File:Move File
[Object] - [N]	Select next	[Control][↓] [Control][F1] - [↓] [Control][F1] - [N]	Not available
[Object] - [R]	Next history/ command	[Control][F1] - [R] [Control][F1] - [V] [Control][Meta][.] [Control][Meta][>]	Not available
[Object] - [S]	Sort object	[Control][F1] - [S]	Tools:Mail:Sort Mailbox
[Object] - [U]	Previous history/ command	[Control][F1] - [U] [Control][Meta][,] [Control][Meta][<]	Tools:Mail:Undelete Mail Message

Table D-2 Object Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Object] - [v]	Next history/ command	[Control][F1] - [R] [Control][F1] - [V] [Control][Meta][.] [Control][Meta][>]	Not available
[Object] - [x]	Release object	[Control][F1] - [X]	File:Close
[Object] - [1] or [Object] - [!]	Expand object	[Meta][1] [Meta][!] [Control][F1] - [1] [Control][F1] - [!]	Not available
[Object] - [.] or [Object] - [>]	Elide object	[Meta][0] [Meta][)] [Control][F1] - [.] [Control][F1] - [>]	Not available
[Object] - [/] or [Object] - [?]	Explain	[F3] [Meta][/ [Meta][?] [Control][F1] - [/] [Control][F1] - [?]	Help:Explain
[Object] - [Tab]	Tab to comment	[Control][F1] - [Tab]	Not available
[Object] - [Promote]	Commit object	[Control][Return] [Control][F1] - [Return]	File:Save
[Object] - [Begin Of]	Select first child	[Control][F1] - [A] [Control][F1] - [B]	Not available
[Object] - [End Of]	Select last child	[Control][F1] - [E]	Not available
[Object] - [↑]	Select previous	[Control][↑] [Control][F1] - [↑]	Not available
[Object] - [↓]	Select next	[Control][↓] [Control][F1] - [↓] [Control][F1] - [N]	Not available
[Object] - [←]	Select parent	[Control][←] [Control][F1] - [←] [Control][F1] - [H] [Control]+d1b click left	Not available
[Object] - [→]	Select child	[Control][→] [Control][F1] - [→] [Control][F1] - [J] [Control]+dbl click right	Not available

REGION OPERATIONS

Table D-3 Region Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Region] - [A]	Beginning of region	[Control][F2] - [A] [Control][F2] - [B]	Navigate:Top of Region
[Region] - [B]	Beginning of region	[Control][F2] - [A] [Control][F2] - [B]	Navigate:Top of Region
[Region] - [C]	Copy region	[Control][F2] - [C] [Control]+click middle or [Control][C] and [Control][Y] in conjunction	Edit:Copy and Edit:Paste in conjunction
[Region] - [D]	Delete region	[Control][W] [Shift][Delete] [Control][F2] - [D] [Control][F2] - [K]	Edit:Cut
[Region] - [E]	End of region	[Control][F2] - [E]	Navigate:Bottom of Region
[Region] - [F]	Fill region	[Control][F2] - [F]	Edit:Fill
[Region] - [H]	Previous region on stack	[Control][F2] - [H] [Control][F2] - [←]	Not available
[Region] - [I]	Next region on stack	[Meta][Y] [Control][F2] - [I] [Control][F2] - [→]	Not available
[Region] - [K]	Delete region	[Control][W] [Shift][Delete] [Control][F2] - [D] [Control][F2] - [K]	Edit:Cut
[Region] - [M]	Move region	[Control][F2] - [M] or [Control][W] and [Control][Y] in conjunction	Edit:Cut and Edit:Paste in conjunction
[Region] - [N]	Push region onto stack	[Control][F2] - [N]	Edit:Copy
[Region] - [P]	Copy top region on stack onto stack	[Control][F2] - [P]	Not available
[Region] - [Q]	Justify region	[Control][F2] - [Q]	Edit:Justify
[Region] - [R]	Move bottom region on stack to top of stack	[Control][F2] - [R]	Not available
[Region] - [T]	Exchange top two regions on stack	[Control][F2] - [T]	Not available
[Region] - [U]	Paste region from top of stack	[Control][Y] [Control][F2] - [U] [Control][F2] - [↑]	Edit:Paste

Table D-3 Region Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Region] - [X]	Deselect region	[Control][V] [Control][F2] - [X]	Edit:Underlines Off
[Region] - [6] or [Region] - [^]	Capitalize region	[Control][F2] - [6] [Control][F2] - [^] [Control][F2] - ['] [Control][F2] - ["]	Edit:Capitalize
[Region] - [9] or [Region] - [()]	Start region	[Control][[] [Control][F2] - [9] [Control][F2] - [() [Control][F2] - [[] [Control][F2] - [{} [Control]+click left	Not available
[Region] - [0] or [Region] - [)]	Finish region	[Control][[] [Control][F2] - [0] [Control][F2] - [)] [Control][F2] - [[] [Control][F2] - [}] [Control]+click right	Not available
[Region] - [-] or [Region] - [_]	Make region into comment	[Control][F2] - [-] [Control][F2] - [_]	Edit:Make into Comment
[Region] - [=] or [Region] - [+]	Uncomment region	[Control][F2] - [=] [Control][F2] - [+]	Edit:Uncomment
[Region] - [[] or [Region] - [{}]	Start region	[Control][[] [Control][F2] - [[] [Control][F2] - [{} [Control][F2] - [9] [Control][F2] - [() [Control]+click left	Not available
[Region] - [)] or [Region] - [}]	Finish region	[Control][[] [Control][F2] - [)] [Control][F2] - [}] [Control][F2] - [0] [Control][F2] - [)] [Control]+click right	Not available
[Region] - ['] or [Region] - ["]	Capitalize region	[Control][F2] - ['] [Control][F2] - ["] [Control][F2] - [6] [Control][F2] - [^]	Edit:Capitalize
[Region] - [.,] or [Region] - [<]	Lowercase region	[Control][F2] - [.,] [Control][F2] - [<]	Edit:Lowercase
[Region] - [.] or [Region] - [>]	Uppercase region	[Control][F2] - [.] [Control][F2] - [>]	Edit:Uppercase
[Region] - [Delete]	Remove top region from stack	[Control][F2] - [Delete]	Not available

Table D-3 Region Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Region] - [Begin Of]	Beginning of region	[Control][F2] - [A] [Control][F2] - [B]	Navigate:Top of Region
[Region] - [End Of]	End of region	[Control][F2] - [E]	Navigate:Bottom of Region
[Region] - [↑]	Paste region from top of stack	[Control][Y] [Control][F2] - [↑] [Control][F2] - [U]	Edit:Paste
[Region] - [↓]	Push region onto stack	[Control][F2] - [↓]	Edit:Copy
[Region] - [←]	Previous region on stack	[Control][F2] - [←] [Control][F2] - [H]	Not available
[Region] - [→]	Next region on stack	[Meta][Y] [Control][F2] - [→] [Control][F2] - [J]	Not available

WINDOW OPERATIONS

Table D-4 Window Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Window] - [A]	Beginning of window	[Control][Meta][Home] [Control][F3] - [A] [Control][F3] - [B]	Not available
[Window] - [B]	Beginning of window	[Control][Meta][Home] [Control][F3] - [A] [Control][F3] - [B]	Not available
[Window] - [C]	Copy window	[Control][F3] - [C]	Copy Window button
[Window] - [D]	Delete window	[Control][Meta][K] [Control][F3] - [D] [Control][F3] - [K] [Control][F3] - [W] [Control][F3] - [X]	Remove Window button
[Window] - [E]	End of window	[Control][Meta][End] [Control][F3] - [E]	Not available
[Window] - [F]	Format windows	[Control][F3] - [F]	Realign Windows button
[Window] - [H]	Enclosing	[F7] [Control][F3] - [H]	Navigate:Enclosing
[Window] - [J]	Join next window	[Control][F3] - [J] [Control][F3] - [O]	Join Next Window button
[Window] - [K]	Delete window	[Control][Meta][K] [Control][F3] - [D] [Control][F3] - [K] [Control][F3] - [W] [Control][F3] - [X]	Remove Window button
[Window] - [M]	Promote window	[Control][Meta][P] [Control][F3] - [M] [Control][F3] - [Z] [Control][F3] - [Return]	Lock Window button
[Window] - [N]	Next window	[Control][Meta][N] [Control][Meta][↓] [Control][F3] - [N] [Control][F3] - [↓]	Not available
[Window] - [O]	Join next window	[Control][F3] - [J] [Control][F3] - [O]	Join Next Window button
[Window] - [T]	Transpose windows	[Control][F3] - [T]	Not available
[Window] - [U]	Previous window	[Control][Meta][U] [Control][Meta][↑] [Control][F3] - [U] [Control][F3] - [↑]	Not available

Table D-4 Window Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Window] - [V]	Child window	[Control][F3] - [V] [Control][F3] - [→]	Not available
[Window] - [W]	Delete window	[Control][Meta][K] [Control][F3] - [D] [Control][F3] - [K] [Control][F3] - [W] [Control][F3] - [X]	Remove Window button
[Window] - [X]	Delete window	[Control][Meta][K] [Control][F3] - [D] [Control][F3] - [K] [Control][F3] - [W] [Control][F3] - [X]	Remove Window button
[Window] - [Y]	Demote window	[Control][Meta][E] [Control][F3] - [Y]	Unlock Window button
[Window] - [Z]	Promote window	[Control][Meta][P] [Control][F3] - [M] [Control][F3] - [Z] [Control][F3] - [Return]	Lock Window button
[Window] - [1] or [Window] - [!]	Expand window	[Control][.] [Control][>] [Control][Meta][!] [Control][F3] - [1] [Control][F3] - [!]	Expand Window button
[Window] - [.] or [Window] - [>]	Elide window	[Control][,] [Control][<] [Control][Meta][,] [Control][F3] - [.] [Control][F3] - [>]	Shrink Window button
[Window] - [/] or [Window] - [?]	Window directory	[Control][/] [Control][?] [Control][F3] - [/] [Control][F3] - [?]	Not available
[Window] - [↑]	Previous window	[Control][Meta][↑] [Control][Meta][U] [Control][F3] - [↑] [Control][F3] - [U]	Not available
[Window] - [↓]	Next window	[Control][Meta][↓] [Control][Meta][N] [Control][F3] - [↓] [Control][F3] - [N]	Not available
[Window] - [←]	Parent window	[Control][F3] - [←]	Not available
[Window] - [→]	Child window	[Control][F3] - [→] [Control][F3] - [V]	Not available
[Window] - [Delete]	Join previous window	[Control][F3] - [Delete]	Join Previous Window button

Table D-4 Window Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Window] - [Promote]	Promote window	[Control][Meta][P] [Control][F3] - [M] [Control][F3] - [Z] [Control][F3] - [Return]	Lock Window button
[Window] - [Begin Of]	Beginning of window	[Control][Meta][Home] [Control][F3] - [A] [Control][F3] - [B]	Not available
[Window] - [End Of]	End of window	[Control][Meta][End] [Control][F3] - [E]	Not available
[Window] - [Demote]	Demote window	[Control][Meta][E] [Control][F3] - [Y]	Unlock Window button
[Window] - [Edit]	Demote window	[Control][Meta][E] [Control][F3] - [Y]	Unlock Window button
[Window] - [Format]	Format windows	[Control][F3] - [F]	Realign Windows button
[Window] - [Definition]	Window directory	[Control][?] [Control][?] [Control][F3] - [?] [Control][F3] - [?]	Not available

IMAGE OPERATIONS

Table D-5 Image Operations

Logical Key	Operation	Physical Key or Mouse Binding	Menu Item or Button
[Image] - [A]	Beginning of image	[Shift][Home] [Shift][Page Up] [Control][F4] - [A] [Control][F4] - [B]	Top of Image button
[Image] - [B]	Beginning of image	[Shift][Home] [Shift][Page Up] [Control][F4] - [A] [Control][F4] - [B]	Top of Image button
[Image] - [E]	End of image	[Shift][End] [Shift][Page Down] [Control][F4] - [E]	Bottom of Image button
[Image] - [F]	Fill mode on	[Control][F4] - [F]	Edit: Typing Modes
[Image] - [H]	Scroll left	[Shift][←] [Control][F4] - [←] [Control][F4] - [H]	Scroll Left button
[Image] - [I]	Insert mode	[Control][F4] - [I]	Edit: Typing Modes
[Image] - [J]	Scroll right	[Shift][→] [Control][F4] - [→] [Control][F4] - [J]	Scroll Right button
[Image] - [N]	Scroll down	[Page Down] [Control][V] [Shift][↓] [Control][F4] - [↓] [Control][F4] - [N]	Scroll Down button
[Image] - [O]	Overwrite mode	[Control][F4] - [O]	Edit: Typing Modes
[Image] - [U]	Scroll up	[Page Up] [Meta][V] [Control][Z] [Shift][↑] [Control][F4] - [↑] [Control][F4] - [U]	Scroll Up button
[Image] - [X]	Fill mode off	[Control][F4] - [X]	Edit: Typing Modes
[Image] - [/] or [Image] - [?]	Find image	Not available	Not available
[Image] - [↑]	Scroll up	[Page Up] [Meta][V] [Control][Z] [Shift][↑] [Control][F4] - [↑] [Control][F4] - [U]	Scroll Up button

Table D-5 Image Operations (continued)

Logical Key	Operation	Physical Key or Mouse Binding	Menu Item or Button
[Image] - [↓]	Scroll down	[Page Down] [Control][V] [Shift][↓] [Control][F4] - [↓] [Control][F4] - [N]	Scroll Down button
[Image] - [←]	Scroll left	[Shift][←] [Control][F4] - [←] [Control][F4] - [H]	Scroll Left button
[Image] - [→]	Scroll right	[Shift][→] [Control][F4] - [→] [Control][F4] - [J]	Scroll Right button
[Image] - [Begin Of]	Beginning of image	[Shift][Home] [Shift][Page Up] [Control][F4] - [A] [Control][F4] - [B]	Top of Image button
[Image] - [End Of]	End of image	[Shift][End] [Shift][Page Down] [Control][F4] - [E]	Bottom of Image button

LINE OPERATIONS

Table D-6 Line Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Line] - [A]	Beginning of line	[Home] [Control][A] [Control][F5] - [A] [Control][F5] - [B]	Not available
[Line] - [B]	Beginning of line	[Home] [Control][A] [Control][F5] - [A] [Control][F5] - [B]	Not available
[Line] - [C]	Copy line	[Control][Meta][C] [Control][F5] - [C]	Not available
[Line] - [D]	Delete line	[Control][Meta][D] [Control][F5] - [D]	Not available
[Line] - [E]	End of line	[End] [Control][E] [Control][F5] - [E]	Not available
[Line] - [I]	Insert line	[Control][F5] - [I]	Not available
[Line] - [J]	Join lines	[Control][F5] - [J]	Not available
[Line] - [K]	Delete to end-of-line	[Control][K] [Control][Delete] [Control][F5] - [K] [Control][F5] - [Delete]	Not available
[Line] - [O]	Open new line	[Control][O] [Control][F5] - [O]	Not available
[Line] - [T]	Transpose lines	[Control][Meta][T] [Control][F5] - [T]	Not available
[Line] - [4] or [Line] - [\$]	Center line	[Control][F5] - [4] [Control][F5] - [\$]	Not available
[Line] - [6] or [Line] - [^]	Capitalize line	[Control][F5] - [6] [Control][F5] - [^]	Not available
[Line] - [.] or [Line] - [<]	Lowercase line	[Control][F5] - [.] [Control][F5] - [<]	Not available
[Line] - [.] or [Line] - [>]	Uppercase line	[Control][F5] - [.] [Control][F5] - [>]	Not available
[Line] - [/?] or [Line] - [?]	Explain line	[Control][Home] [Control][F5] - [/?] [Control][F5] - [?]	Not available
[Line] - [↑]	Line up	[↑] [Control][U]	Not available

Table D-6 Line Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Line] - [↓]	Line down	[↓] [Control][N]	Not available
[Line] - [Delete]	Delete to end-of-line	[Control][K] [Control][Delete] [Control][F5] - [K] [Control][F5] - [Delete]	Not available
[Line] - [Begin Of]	Beginning of line	[Home] [Control][A] [Control][F5] - [A] [Control][F5] - [B]	Not available
[Line] - [End Of]	End of line	[End] [Control][E] [Control][F5] - [E]	Not available

WORD OPERATIONS

Table D-7 Word Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Word] - [A]	Beginning of word	[Control][F6] - [A] [Control][F6] - [B]	Not available
[Word] - [B]	Beginning of word	[Control][F6] - [A] [Control][F6] - [B]	Not available
[Word] - [D]	Delete word	[Meta][D] [Control][F6] - [D]	Not available
[Word] - [E]	End of word	[Control][F6] - [E]	Not available
[Word] - [I]	Speller learn word	[Control][F6] - [I]	Edit:Speller:Learn Word
[Word] - [J]	Next word	[Meta][J] [Meta][→] [Control][F6] - [J] [Control][F6] - [→]	Not available
[Word] - [K]	Delete to end-of-word	[Meta][K] [Control][F6] - [K] [Control][F6] - [Delete]	Not available
[Word] - [M]	Speller check image	[Control][F6] - [M]	Edit:Speller:Check Image
[Word] - [N]	Speller explain next	[Control][F6] - [N]	Edit:Speller:Explain Next
[Word] - [R]	Speller learn replacement	[Control][F6] - [R]	Edit:Speller:Learn Replacement
[Word] - [T]	Transpose words	[Meta][T] [Control][F6] - [T]	Not available
[Word] - [W]	Speller window	[Control][F6] - [W]	Edit:Speller:Speller Window
[Word] - [X]	Speller exchange word	[Control][F6] - [X]	Edit:Speller:Exchange Word
[Word] - [6] or [Word] - [^]	Capitalize word	[Meta][6] [Meta][^] [Control][F6] - [6] [Control][F6] - [^]	Not available
[Word] - [,] or [Word] - [<]	Lowercase word	[Meta][,] [Meta][<] [Control][F6] - [,] [Control][F6] - [<]	Not available
[Word] - [.] or [Word] - [>]	Uppercase word	[Meta][.] [Meta][>] [Control][F6] - [.] [Control][F6] - [>]	Not available
[Word] - [/] or [Word] - [?]	Speller check text	[Control][F6] - [/] [Control][F6] - [?]	Edit:Speller:Check Text

Table D-7 Word Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Word] - [↑]	Not bound	Not applicable	Not applicable
[Word] - [↓]	Speller explain next	[Control][F6] - [↓]	Edit:Speller:Explain Next
[Word] - [←]	Previous word	[Meta][←] [Control][F6] - [←]	Not available
[Word] - [→]	Next word	[Meta][→] [Meta][J] [Control][F6] - [→] [Control][F6] - [J]	Not available
[Word] - [Begin Of]	Beginning of word	[Control][F6] - [A] [Control][F6] - [B]	Not available
[Word] - [End Of]	End of word	[Control][F6] - [E]	Not available
[Word] - [Delete]	Delete to end-of-word	[Meta][K] [Control][F6] - [K] [Control][F6] - [Delete]	Not available

MARK OPERATIONS

Table D-8 Mark Operations

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Mark] - [A]	Start macro	[Meta][[Control][F7] - [A] [Control][F7] - [B] [Control][F7] - [9] [Control][F7] - [0] [Control][F7] - [] [Control][F7] - []	Tools:Macro:Begin Macro Def
[Mark] - [B]	Start macro	[Meta][[Control][F7] - [A] [Control][F7] - [B] [Control][F7] - [9] [Control][F7] - [0] [Control][F7] - [] [Control][F7] - []	Tools:Macro:Begin Macro Def
[Mark] - [E]	Finish macro	[Meta][[Control][F7] - [E] [Control][F7] - [0] [Control][F7] - [D] [Control][F7] - [] [Control][F7] - []	Tools:Macro:End Macro Def
[Mark] - [F]	Bind macro to key	[Control][Meta][=] [Control][Meta][+] [Control][F7] - [F]	Tools:Macro:Bind Macro to Key
[Mark] - [H]	Previous mark	[Control][Meta][M] [Control][F7] - [H] [Control][F7] - [←]	Not available
[Mark] - [J]	Next mark	[Control][F7] - [J] [Control][F7] - [→]	Not available
[Mark] - [M]	Execute macro	[Meta][X] [Control][F7] - [M] [Control][F7] - [X] [Control][F7] - [Return]	Tools:Macro:Execute Macro
[Mark] - [N]	Push mark	[Control][M] [Control][@] [Control][F7] - [N] [Control][F7] - [↓]	Not available
[Mark] - [P]	Copy top mark	[Control][F7] - [P]	Not available
[Mark] - [R]	Rotate top marks	[Control][F7] - [R]	Not available
[Mark] - [T]	Swap marks	[Control][F7] - [T]	Not available
[Mark] - [U]	Top mark	[Control][F7] - [U] [Control][F7] - [↑]	Not available

Table D-8 Mark Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Mark] - [X]	Execute macro	[Meta][X] [Control][F7] - [M] [Control][F7] - [X] [Control][F7] - [Return]	Tools:Macro:Execute Macro
[Mark] - [9] or [Mark] - [()]	Start macro	[Meta][[] [Control][F7] - [A] [Control][F7] - [B] [Control][F7] - [9] [Control][F7] - [() [Control][F7] - [[] [Control][F7] - [{}]	Tools:Macro:Begin Macro Def
[Mark] - [0] or [Mark] - [)]	Finish macro	[Meta][[] [Control][F7] - [E] [Control][F7] - [0] [Control][F7] - [)] [Control][F7] - [[] [Control][F7] - [}]	Tools:Macro:End Macro Def
[Mark] - [[] or [Mark] - [{}]	Start macro	[Meta][[] [Control][F7] - [A] [Control][F7] - [B] [Control][F7] - [9] [Control][F7] - [() [Control][F7] - [[] [Control][F7] - [{}]	Tools:Macro:Begin Macro Def
[Mark] - [}] or [Mark] - [)]	Finish macro	[Meta][[] [Control][F7] - [E] [Control][F7] - [0] [Control][F7] - [)] [Control][F7] - [[] [Control][F7] - [}]	Tools:Macro:End Macro Def
[Mark] - [↑]	Top mark	[Control][F7] - [↑] [Control][F7] - [U]	Not available
[Mark] - [↓]	Push mark	[Control][M] [Control][@] [Control][F7] - [↓] [Control][F7] - [N]	Not available
[Mark] - [←]	Previous mark	[Control][Meta][M] [Control][F7] - [←] [Control][F7] - [H]	Not available
[Mark] - [→]	Next mark	[Control][F7] - [→] [Control][F7] - [J]	Not available

Table D-8 Mark Operations (continued)

Logical Key	Operation	Physical Key or Mouse Button	Menu Item or Button
[Mark] - [Begin Of]	Start mark	[Meta][[]] [Control][F7] - [A] [Control][F7] - [B] [Control][F7] - [9] [Control][F7] - [({] [Control][F7] - [[]] [Control][F7] - [}]	Tools:Macro:Begin Macro Def
[Mark] - [End Of]	Finish mark	[Meta][[]] [Control][F7] - [E] [Control][F7] - [0] [Control][F7] - [)] [Control][F7] - [}] [Control][F7] - [}]	Tools:Macro:End Macro Def
[Mark] - [Delete]	Delete mark	[Control][F7] - [Delete]	Not available
[Mark] - [Promote]	Execute macro	[Meta][X] [Control][F7] - [M] [Control][F7] - [X] [Control][F] - [Return]	Tools:Macro:Execute Macro

Special Keys					
[Key]	Unmodified	[Shift][Key]	[Control][Key]	[Meta][Key]	[Control][Meta][Key]
[Return]	Return	Return	Commit	Debug	
[Tab]	Go to next tab stop	Go to next tab stop			Show tabs
[Delete] or [Back Space]	Delete previous character	Delete region	Delete to end of line	Delete previous word	Delete white space*
[↑]	Cursor up	Scroll up	Select previous	Previous item	Previous window
[↓]	Cursor down	Scroll down	Select next	Next item	Next window
[←]	Cursor left	Scroll left	Select first/parent	Previous word	Enclosing in place
[→]	Cursor right	Scroll right	Select last/child	Next word	Definition in place (body)
[Home]	Beginning of line	Top of image	Line information		Top of window
[End]	End of line	Bottom of image	End input		Bottom of window
[Page Up]	Scroll up	Top of image			
[Page Down]	Scroll down	Bottom of image			

* **Caution:** This operation kills the window manager on an IBM RS/6000.

Mouse			
Action	Left Button	Middle Button	Right Button
Click	Position cursor; start Motif selection	Copy Motif selection	
[Shift] + click	End Motif selection		
Double click	Definition		Enclosing
[Shift] + double click	Definition in place		Enclosing in place
[Control] + click	Start region	Copy region	Finish region
[Control] + double click	Select object/parent		Select child
Drag	Motif selection		
[Control] + drag	Region selection		

Moving within Any Image	
Operation	Key or Key Combination*
Cursor up	[↑], [Control][U]
Cursor down	[↓], [Control][N]
Cursor left	[←], [Control][H], [Control][F]
Cursor right	[→], [Control][J], [Control][B]
Next word	[Meta][→], [Meta][J]
Previous word	[Meta][←]
Next item	[F12], [Meta][↓], [Meta][N]
Previous item	[Shift][F12], [Meta][↑], [Meta][U]
Next underline	[Meta][=], [Meta][+]
Previous underline	[Meta][-], [Meta][_]
Beginning of line	[Home], [Control][A]
End of line	[End], [Control][E]
Top of image	[Shift][Home], [Shift][Page Up]
Bottom of image	[Shift][Home], [Shift][Page Down]
Top of window	[Control][Meta][Home]
Bottom of window	[Control][Meta][End]
Scroll up	[Page Up], [Shift][↑], [Control][Z], [Meta][V]
Scroll down	[Page Down], [Shift][↓], [Control][V]
Scroll left	[Shift][←]
Scroll right	[Shift][→]
Push mark	[Control][M], [Control][@]
Go to mark	[Control][Meta][M]
Expand image	[Meta][!], [Meta][!]
Elide image	[Meta][0], [Meta][!]

* Left mouse button positions cursor at the location of the pointer.

Basic Editing Operations	
Operation	Key or Key Combination
Create text file	[F12]
Open file	[F2]
Delete previous character	[Delete], [Back Space]
Delete next character	[Control][D]
Delete previous word	[Meta][Delete]
Delete next word	[Meta][D]
Delete to end of word	[Meta][K]
Delete line	[Control][Meta][D]
Delete to beginning of line	[Meta][Delete], [Meta][Back Space]
Delete to end of line	[Control][Delete], [Control][Back Space], [Control][K]
Delete white space	[Control][Meta][Delete], [Control][Meta][Back Space]*
Transpose characters	[Control][T]
Transpose words	[Meta][T]
Transpose lines	[Control][Meta][T]
Copy line	[Control][Meta][C]
Open new line	[Control][O]
Go to next tab stop	[Tab], [Shift][Tab]
Lowercase word	[Meta][.], [Meta][<]
Uppercase word	[Meta][.], [Meta][>]
Capitalize word	[Meta][@], [Meta][^]
Quote next character	[Control]['], [Control][*]
Search previous	[Control][R]
Search next	[Control][S]
Replace previous	[Control][Meta][R]
Replace next	[Control][Meta][S]

* Caution: This operation kills the window manager on an IBM RS/6000.

Traversing the Environment*		
Operation	Key or Key Combination	Mouse Operation
Definition	[F5]	Double click left
Definition in place (spec)	[Shift][F5]	[Shift] + double click left
Definition in place (body)	[Control][Meta][→]	
Enclosing	[F7]	Double click right
Enclosing in place	[Shift][F7], [Control][Meta][←]	[Shift] + double click right
Home library	[Shift][F10]	

* Environment traversal operations are also available from the File and Navigate menus.

Getting Help and Other Information*	
Operation	Key or Key Combination
Explain	[F3], [Meta][?], [Meta][?]
Help on command	[F1]
Help on key	[Control][Q]
Prompt for key	[Shift][F1]
Show tabs	[Control][Meta][Tab]
Line information	[Control][Home]

* Help is also available through the Help menu and by clicking on the Help button in dialog boxes.

Managing Windows*	
Operation	Key or Key Combination
Window directory	[Control][/], [Control][?]
Next window	[Control][Meta][↓], [Control][Meta][N]
Previous window	[Control][Meta][↑], [Control][Meta][U]
Shrink window	[Control][.], [Control][<], [Control][Meta][)]
Expand window	[Control][.], [Control][>], [Control][Meta][[
Lock window	[Control][Meta][P]
Unlock window	[Control][Meta][E]
Kill window	[Control][Meta][K]
Repaint Environment area	[Control][L]
Clear Environment area	[Meta][L]

* Windows also can be managed using the window-control buttons at the top of the Access window.

Region and Selection Operations		
Operation	Key or Key Combination	Mouse Operation
Start region	[Control][I]	[Control] + click left, [Control] + drag
Finish region	[Control][I]	[Control] + click right
Cut region	[Control][W], [Shift][Delete], [Shift][Back Space]	
Copy Region	[Control][C] (to buffer)	[Control] + click middle (and paste)
Paste region	[Control][Y]	
Paste next region	[Meta][Y]	
Deselect region	[Control][X], [Control][\]	
Select previous item	[Control][↑]	
Select next item	[Control][↓]	
Select first/parent	[Control][←]	[Control] + click left
Select last/child	[Control][→]	[Control] + click right
Copy Motif selection		Click middle
End Motif selection		[Shift] + click left

Writing Ada Programs	
Operation	Key or Key Combination
Create Ada unit	[Shift][F11]
Save	[Control][Return], [Shift][Enter]
Promote	[F8]
Demote	[Shift][F8]
Debug	[Meta][Return]
Edit unit or construct	[Shift][F2]
Complete	[Shift][F6]
Format	[F9]
Semanticize	[Shift][F9]
Explain	[F3], [Meta][/], [Meta][?]
Underlines off	[Shift][F3]
Insert =>	[Control][=], [Control][+]
Insert :=	[Control][:;], [Control][:]
Insert --	[Control][]
Insert RDF annotation*	[Meta][2], [Meta][@]
Insert ("	[Control][(]
Insert ")	[Control][)]

* Key struck after [Meta][@] or [Meta][2] determines the particular RDF annotation inserted.

Index

A

Accept Changes dialog box	100
accepting changes.	99
Access	
as an X application	126
commands provided	5
compatibility with layered products	xvi
configurations for	125
controlling jobs	111
creating and modifying text files	63
customizing your workspace	117
debugging	53
defined	1
editing text	67
equivalents to Environment commands	137
equivalents to key and mouse bindings.	151
getting help	19
getting started	1
how it works	125
logging in through Access	1
logging into the Environment	2
logging out	11
main window.	1, 3
managing Environment windows	27
managing libraries	77
opening an Access window	2
performing operations in windows	5
requirements for running.	127
setting up	125
traversing Ada programs	37
traversing the Environment	35
user-interface basics	129
using CMVC	87
using special features	13
Debugger Palette	16, 53
Function Key Palette	16
Image Palette	15
Just-Do-It mode	18
user-defined buttons.	14
window-control buttons	13

Access (*continued*)

- version you are using, help on. 21
- writing Ada programs 39

Access commands

- equivalents to Environment commands. 137

Access help window 19

Access Server 126

Access window

- see* windows, Access

Access X Client 126

activity

- adding an entry 104
- creating 104
- defined 104

Ada constructs, showing unused. 38

Ada names, showing occurrences 37

Ada programs

- creating loaded main program 51
- debugging 53
- executing 50
- traversing 37
- writing 39

Ada specifications

- displaying 25
- traversing 37

Ada subprograms

- adding to package 49
- making body into a subunit. 50

Ada units

- creating 39, 40
- demoting 43
- making subunit in-line with parent unit. 50
- modifying 45
 - adding to 46
 - changing incrementally. 47
 - changing name or kind. 48
 - deleting part of 48
- moving between specification and body 37
- promoting 41
- saving incomplete 50
- selecting parent/child items 45
- states. 45
- viewing parent 37

B

- background, putting job in. 112
- bell, visual, setting. 120

breakpoints 57
removing 57
setting 57
showing 57
Browse dialog box	35, 64
browsing, <i>see</i> traversing	

C

call stack 58
displaying 58
displaying parameters for frame 59
displaying source for frame 59
traversing from 59
case of text, changing 74
check box, in dialog boxes	133
Check In dialog box 99
Check Out dialog box 98
checking out/in an object for changes 98
CMVC	
accepting changes 99
checking out/in an object for changes 98
collecting information about controlled objects	105
creating a new activity	104
creating a subsystem 87
creating a system 96
creating a venture	107
creating a work order	105
creating a work-order list	106
getting information about a view	107
getting the history of an object	108
joining objects in different views	101
making objects controlled or uncontrolled 97
releasing configurations 92
reverting to a previous generation	103
severing objects in different views	102
starting the CMVC editor	105
using 87
CMVC editor, starting	105
CMVC menu 4
coded state 45
command windows 8
creating and executing a command-window program	121
entering a new command	123
executing commands from 9
getting command completion	121
getting the parameters of a command bound to a key	124
going back to previous commands	123
moving in	122

command window (<i>continued</i>)	
reexecuting a command	123
using	121
commands	
Access	5
Environment	5, 9
Access equivalents	137
getting help on	24
getting list of	22
executing from command windows	9
executing from menus	7, 129
with user-defined buttons	117
executing with mouse	6
item operation	
executing	10
configuration management	87
<i>see also</i> CMVC	
configurations	
for setting up Access	125
releasing	92
Control/Uncontrol dialog box	97
controlling	
jobs	111
objects	97
conventions	
mouse	xvi
text	xv
windows	xv
Copy File dialog box	81
copying	
objects	81
text	70
customizing your Access workspace	117

D

Debug menu	4
Debugger Palette	16
closing	54
displaying	53
using	53
debugging	53
continuing a task	56
displaying program being debugged	55
displaying value of program variable	58
examining call stack	58
exception handling	60
modifying variable values	58
redisplaying the Environment's debugger window	54
showing information	61

debugging (<i>continued</i>)	
starting the debugger 54
stepping through a program 55
stopping a task 56
stopping the debugger 55
using breakpoints 57
using the Debugger Palette 53
default button, in dialog boxes132, 135
Delete File Confirmation dialog box 80
Delete File dialog box 80
deleting text 73
Demote dialog box 44
destroying objects 80
dialog boxes. 98
Accept Changes 100
Browse	35, 64
Check In 99
Check Out 98
Control/Uncontrol 97
Copy File 81
Delete File 80
Delete File Confirmation 80
Demote 44
elements in 132
File Close	30, 76
File Revert 65
Job Connect 113
Job Enable/Disable/Kill 113, 114, 115
Join Controlled CMVC Objects 102
Load Main Program. 51
Move File 82
navigating 134
New Activity 104
New Code View 96
New Directory 79
New Release View	93, 95
New Spec View 91
New Subsystem 88
New System 97
New Text File 63
New Venture 107
New Work Order 106
New Work Order List 106
New Working View 89
New World 79
Object History Information 108
OK 11
Print 83
Promote 42
Rational Access Error Message 18
Rational Environment Help 22
responding to. 131
using keyboard 134
using mouse 133

dialog boxes (<i>continued</i>)	
Revert to Generation	103
Run Program	51
Search and Replace	71
Set Window Frames	31
Sever Controlled CMVC Objects	103
shortcut for canceling	136
summary of keys for responding to	136
terms for describing	132
Typing Modes	66
directories	
creating	79
defined	77

E

-e option	2
Edit menu	3
editing text	67
changing case	74
copying	70
deleting	73
filling a region	75
getting line information	75
justifying a region	75
moving Environment cursor with keyboard	67
moving text	70
saving changes	76
searching for and replacing	71
selecting text	68
in the Environment	68
with Motif	69
transposing	73
entry box, in dialog boxes	133
summary of keys for editing	136
Environment area, of Access window	4, 7
command window	8
Environment cursor	8
Environment windows	7
banner in	7
Environment-window frame	8
message window	7
message-window banner	7
mouse pointer	9
Environment commands	9
Access equivalents	137
getting help on	24
Environment cursor	8
moving with keyboard	67
Environment packages	
getting list of	22
<i>see also</i> packages	

Environment windows	7
banner in	7
symbols used	8
changing size	28
changing size of user area and full image list	33
closing the Image Palette	33
displaying the Image Palette	31
finding windows with uncommitted changes	33
frame of	8
getting list of	31
locking and unlocking	30
managing	27
moving between	27
moving within	27
moving cursor	28
traversing using a mark	28
redisplaying from the Image Palette	32
removing	30
restoring	34
saving	33, 34
searching for	32
setting number of frames	31
setting up a standard set	33
splitting	29
updating the Image Palette	33
<i>see also</i> windows, Environment	
Environment, <i>see</i> Rational Environment	
errors, getting help on	25
exception handling	60
catching exceptions	60
returning to point of program suspension	61
showing exceptions	61

F

File Close dialog box	30, 76
File menu	3
File Revert dialog box	65
files	
creating	63
opening existing for editing	64
reverting to previous version	65
saving	64
setting tabs	65
setting typing modes	65
viewing	63
fill mode, setting	66
Filter button	22
foreground, putting job in	113
Function Key Palette	10, 16

function keys	
getting help	21

G

-geometry option	2
----------------------------	---

H

help	
displaying Ada specifications	25
getting	19
getting list of topics	22
on Access information.	20
on Environment commands.	24
on errors	25
on key bindings.	21
on keys	21
on menus	20
on mouse	20
on online help system.	20
on version of Access	21
on window-control buttons	20
Help menu	4
help topics, getting list	22
help windows	19
using.	19

I

-icontitle option	2
identifiers, getting definition	38
image operations	163
Image Palette	15, 31
closing	33
displaying	31
updating	33
images	
printing	83
insert mode, setting	66
installed state	45
inverse video, changing screen to	120
item keys	10

item-operation commands	
executing	10

J

Job Connect dialog box	113
Job Enable/Disable/Kill dialog box	113, 114, 115
job identification number	111
jobs	
controlling	111
defined	111
disabling	113
disconnecting from (putting in background)	112
displaying current	111
enabling	114
reconnecting to (putting in foreground)	113
Join Controlled CMVC Objects dialog box	102
Just-Do-It mode	18

K

key bindings	
getting help	21
keyboard focus	134
keys	
Access equivalents to logical key bindings.	151
binding macros to	119
function	
getting help.	21
getting help	21
item	10
logical	152
operation	10
rebinding	120
summary of for editing text-entry boxes	136
summary of for responding to dialog boxes	136

L

label, in dialog boxes	132
libraries	
controlling display	77
copying objects	81
creating nonsubsystem libraries	78
defined	77
destroying objects	80

libraries (<i>continued</i>)	
managing	77
moving or renaming objects	82
printing objects and images	83
traversing	35
library-level programs, executing	50
line operations	165
list box, in dialog boxes	132
Load Main Program dialog box	51
loaded main program, creating	51
logging in through Access	1
logging into the Environment	2
logging out from Access	11
logical keys, fundamental set and Access equivalents	152

M

macros, building and executing	119
managing	
Environment windows	27
libraries	77
mark	
making	28
traversing to	28
mark operations	169
menu bar	3
menus	
choosing commands from	129
CMVC	4
Debug	4
Edit	3
executing commands from	7
with keyboard	130
with mouse	129
with user-defined buttons	117
with window-control buttons	131
File	3
getting help	20
Help	4
Navigate	3
Program	4
terms for describing menu commands	129
Tools	4
message window	7
banner in	7
mnemonics	130

mouse	
Access equivalents to mouse buttons	151
executing commands	6
from menus	129
getting help	20
summary of operations in menus	130
terminology related to	xvi
using to respond to dialog boxes	133
mouse pointer	9
Move File dialog box	82
moving	
Environment cursor	67
objects	82
text	70

N

Navigate menu	3
New Activity dialog box	104
New Code View dialog box	96
New Directory dialog box	79
New Release View dialog box	93, 95
New Spec View dialog box	91
New Subsystem dialog box	88
New System dialog box	97
New Text File dialog box	63
New Venture dialog box	107
New Work Order dialog box	106
New Work Order List dialog box	106
New Working View dialog box	89
New World dialog box	79

O

Object History Information dialog box	108
object operations	155
objects	
accepting changes from	101
checking out/in for changes	98
collecting information about controlled	105
copying	81

- objects (*continued*)
 - destroying 80
 - getting history of 108
 - joining in different views 101
 - making controlled or uncontrolled 97
 - moving or renaming 82
 - printing 83
 - severing in different views 102
- OK dialog box 11
- online help 19
 - see also* help
- operation keys 10
- option button, in dialog boxes 133
- option menu, in dialog boxes 133
- OSF/Motif-based user interfaces xvii, 1
- overwrite mode, setting 65

P

- packages
 - adding subprogram to 49
 - getting list of 22
 - making body into a subunit 50
 - viewing specification 38
- pattern-matching wildcards 24
- popup menu, in dialog boxes 133
- Print dialog box 83
- printing objects and images 83
 - format options 85
 - other options 85
 - page layout options 85
 - selecting the printer 84
 - specifying the pages to print 84
- Program menu 4
- project partitioning 87
 - see also* CMVC
- Promote dialog box 42
- prompt
 - moving to 122
 - turning off 122

R

- radio button, in dialog boxes 133
- Rational Access Error Message dialog box 18

Rational Access, <i>see</i> Access	
rational command	2
Rational Environment	
getting help on	22
logging in through Access	1, 2
traversing	35
using command windows	121
Rational Environment Help dialog box	22
region operations	157
releasing configurations	92
renaming objects	82
Revert to Generation dialog box	103
reverting to a previous generation	103
Run Program dialog box	51

S

sash.	4
sash control	4
saving	
changes to text	76
current macros	119
files	64
incomplete Ada units	50
set of Environment windows	34
user-defined buttons	118
scale bar, in dialog boxes	133
screen, changing to inverse video	120
scroll arrow, in dialog boxes	132
scroll bar, in dialog boxes	132
Search and Replace dialog box	71
searching for and replacing text	71
options	72
selecting text	68
in the Environment	68
with Motif	69
Set Window Frames dialog box	31
Sever Controlled CMVC Objects dialog box	103
severing objects in different views	102
slider, in dialog boxes	132
source state	45
stack frame	58

stepping through a program 55
subprograms	
adding to package 49
making body into a subunit 50
subsystems	
creating 87
making a path 89
making a spec view 91
making a subpath 90
defined 87
symbols	
in banner of Environment windows 8
wildcards 24
systems	
creating 96
defined 96

T

tabs, setting 65
text conventions xv
text files, <i>see</i> files	
text, editing, <i>see</i> editing text	
text-entry box, in dialog boxes 133
summary of keys for editing 136
-title option 2
Tools menu 4
transposing text 73
traversing	
Ada programs 37
Environment 35
from the call stack 59
Typing Modes dialog box 66
typing modes, setting 65

U

underline	
moving to 122
turning off 122
user interface, basic terms and operations 129
user-defined buttons 4, 14
activating 118
changing size of button area 118

user-defined buttons (<i>continued</i>)	
creating for a menu command	118
deleting	118
executing menu commands	117
saving	118

V

venture	
creating	107
defined	107
version control	87
<i>see also</i> CMVC	
version of Access, help on	21
view	
accepting changes from	99
defined	87
getting information about	107
joining objects in different	101
making a code view	96
making a configuration release.	94
making a release view.	92
making a spec view	91
severing objects in different.	102
visual bell, setting	120

W

wildcards, pattern matching	24
window operations	160
window-control buttons.	4
getting help	20
movement with	27
using.	13
windows	
Access	1, 3
control buttons	4, 13
Environment area.	4, 7
help	19
menu bar	3
opening	2
performing operations in	5
sash	4
sash control.	4
user-defined buttons.	4, 14
command	8, 121
executing commands from	9
<i>see also</i> command windows	

windows (<i>continued</i>)	
Environment	7
<i>see also</i> Environment windows	
message	7
terminology for	xv
word operations	167
wordwrap column, changing	66
wordwrap, setting	66
work order	
creating	105
defined	105
work-order list	
creating	106
defined	106
workspace, customizing	117
worlds	
creating	79
defined	77

X

.Xdefaults file	1
X Window System.	xvii, 1
application components	125

RATIONAL

READER'S COMMENTS

Note: This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

Indicate any additions or changes you would like to see in the index.

How much experience have you had with the Rational Environment?

- 6 mo. or less 6 mo.–1 year 1–3 years 3 years or more

How much experience have you had with the Ada programming language?

- 6 mo. or less 6 mo.–1 year 1–3 years 3 years or more

Name (optional) _____ Date _____

Company _____

Address _____

City _____ State _____ ZIP Code _____

Please return this form to: **Publications Department**
RATIONAL
3320 Scott Boulevard
Santa Clara, CA 95054-3197

RATIONAL

READER'S COMMENTS

Note: This form is for documentation comments only. You can also submit problem reports and comments electronically by using the SIMS problem-reporting system. If you use SIMS to submit documentation comments, please indicate the manual name, book name, and page number.

Did you find this book understandable, usable, and well organized? Please comment and list any suggestions for improvement.

If you found errors in this book, please specify the error and the page number. If you prefer, attach a photocopy with the error marked.

Indicate any additions or changes you would like to see in the index.

How much experience have you had with the Rational Environment?

- 6 mo. or less 6 mo.–1 year 1–3 years 3 years or more

How much experience have you had with the Ada programming language?

- 6 mo. or less 6 mo.–1 year 1–3 years 3 years or more

Name (optional) _____ Date _____

Company _____

Address _____

City _____ State _____ ZIP Code _____

Please return this form to: **Publications Department**
RATIONAL
3320 Scott Boulevard
Santa Clara, CA 95054-3197