Rational Environment
Basic Operations

Rational Terminal

Document Control Number: 8001A-03 (803-002318)
Rev. 4.0, November 1985
Rev. 4.1, December 1985
Rev. 4.2, March 1986
Rev. 4.3, July 1986
Rev. 5.0, July 1987 (Delta)

This document subject to change without notice.

Note the Reader's Comments form on the last page of this book, which requests the user's evaluation to assist Rational in preparing future documentation.

Rational
1501 Salado Drive
Mountain View, California 94043

Contents

Chapter 1. Logging In and Out 1
- Logging In 1
- Logging Out 1
- Saving Changes 2

Chapter 2. Getting Help 3
- Getting Help on Help 3
- Getting Help on a Specific Item 3
- Getting Help on Keys 4
- Displaying Ada Specifications 4
- Displaying the Help Window 4
- Getting Help on Errors 4

Chapter 3. Executing Commands 5
- Creating and Executing a Command Window Program 5
- Expanding a Command Window 5
- Shrinking a Command Window 5
- Getting Command Completion 6
- Moving to the Next Prompt or Underline 6
- Moving to the Previous Prompt or Underline 6
- Turning Off a Prompt 6
- Reexecuting the Same Command 6
- Changing and Reexecuting a Command 7
- Entering a New Command in the Same Command Window 7
- Clearing a Command Window of Unneeded Text 7
- Going Back to Previous Commands 7
- Getting the Parameters of a Command Bound to a Key 7

Chapter 4. Managing Windows 9
Finding a Window Using the Window Directory 9
Deleting Windows from the Window Directory 9
Moving between Windows 9
Expanding a Window 10
Shrinking a Window 10
Expanding Current Window to Include Next Frame 10
Expanding Current Window to Include Previous Frame 10
Transposing Windows 10
Realigning the Windows on the Screen 11
Removing a Window 11
Locking a Window on the Screen 11
Unlocking a Window on the Screen 12
Scrolling the Image 12

Chapter 5. Traversing the Environment 13
Viewing a Library 13
Viewing an Object in a Library 13
Viewing a Library's Parent 13
Viewing Your Home Library 13
Viewing the Specification of an Environment Package 14

Chapter 6. Using General Editing Operations 15
Selecting an Arbitrary Region of Text 15
Moving Selected Text 15
Copying Selected Text 15
Searching for a String 16
Searching and Replacing a String 16
Searching and Replacing All Occurrences of a String 17
Deleting Text 17
Joining Lines 17
Transposing Text 18
Changing the Case of Text 19

Chapter 7. Writing Text Files 21
Creating a File 21
Viewing a File 21
Editing an Existing File 21
Saving a File 22
Setting Tabs 22
Setting Overwrite Mode On 22
Setting Insert Mode On 23
Setting Wordwrap for Text 23
Changing the Wordwrap Column 23
Turning Wordwrap Off 23

Chapter 8. Writing Ada Programs 25
Creating an Ada Package Specification 25
Creating an Ada Package Body 26
Creating an Ada Subprogram 27
Creating a Subunit 27
Importing Units 27
Adding a Statement, Declaration, or Comment 28
Changing a Statement, Declaration, or Comment 29
Deleting a Statement, Declaration, or Comment 30
Changing the Name or Kind of an Ada Unit 31
Adding a Subprogram to a Package 32
Making a Package Body or Subprogram Body into a Subunit 34
Making a Subunit In-line in the Parent 34
Demoting a Unit and Its Dependents 34
Making a Library Program Executable 34
Executing a Library Program 35
Saving the Changes of Incomplete Units 35
Setting Overwrite Mode On 35
Setting Insert Mode On 35

Chapter 9. Browsing Ada Programs 37
Getting the Definition or Use of an Identifier 37
Viewing the Specification of an Environment Package 37
Viewing a Unit's Specification from Its Body 37
Viewing a Unit's Body from Its Specification 38
Viewing a Unit's Parent 38
Showing the Using Occurrences of a Defined Ada Name 38

Chapter 10. Debugging 39
Starting the Debugger 39
Stopping the Debugger 39
Displaying the Program Being Debugged 39
Displaying the Value of a Program Variable 39
Displaying the Call Stack 40
Displaying Source for a Call Stack Frame 40
Displaying Parameters for a Call Stack Frame 40
Stepping Through the Program 40
Executing the Program 41
Setting Up Exception Handling 41
Setting Breakpoints 41
Showing Breakpoints 41
Removing Breakpoints 42
Modifying a Program Variable 42
Returning to the Point of Program Suspension 42
Displaying the Debugger Window 42

Chapter 11. Managing Libraries 43
Controlling the Library Display 43
Creating Libraries 44
Deleting Objects in a Library 44
Undeleting Objects or Previous Versions in a Library 45
Copying Objects in a Library 45
Moving Objects in a Library 46
Renaming Objects in a Library 46
Printing Objects Contained in a Library 47

Chapter 12. Managing Links 49
Listing Links—Simple Method 49
Adding Links—Simple Method 49
Getting the Pathname for an Environment Package 49
Editing Links for a World 50
Controlling the Link Display 50
Inserting a New Link 50
Deleting a Link 51
Viewing the Source of a Link 51
Exiting from the Link Display 51
Adding a Set of Links 51

Replacing a Link 51

Chapter 13. Managing Session Switches 53
Editing Session Switches 53
Controlling the Session Switch Display 53
Modifying Session Switch Values 54
Getting Help on Session Switches 55
Saving Session Switches 55
Exiting from the Session Switch Display 55

Chapter 14. Managing Searchlists 57
Editing the Searchlist for a Session 57
Adding a Component to a Searchlist 57
Deleting a Component from a Searchlist 57
Replacing One Component with Another 58
Viewing the Library Named by a Searchlist Entry 58
Exiting from the Searchlist Display 58

Chapter 15. Managing Jobs 59
Disconnecting from a Job 59
Reconnecting to a Job 59
Killing the Current Job or the Last Job Created 59
Killing Any Job 60

Chapter 16. Customizing Your Workspace 61
Building Macros 61
Defining Your Own Login Procedure 62
Rebinding Keys 62

Chapter 17. Using CMVC 63
Creating a Subsystem 63
Adding, Changing, or Deleting Ada Units in a View 63
Making Ada Units Controlled 63
Making a Subpath 64
Checking Out a Unit for Changes 64
Checking In a Unit after Changes 64
Making a Frozen Release 65
Accepting Changes 65
Getting Information 66

Chapter 18. Networking 67
Logging Into Another System with Telnet 67
Interrupting a Telnet Session 67
Resuming a Telnet Session 68
Terminating a Telnet Session 68
Copying a Single Object or Library onto Another R1000 69
Copying Objects or Libraries from Another R1000 70
Copying Objects onto a Non-R1000 System 71
Copying Objects from a Non-R1000 System 71

Preface

This *Rational Environment Basic Operations* manual describes, with simple step-by-step procedures, how to perform various common operations in the Rational Environment™ using the Rational Terminal.

Not intended as a self-study guide, this manual assumes some familiarity with the Environment. No conceptual discussions are included. Familiarity typically is acquired through the Rational Environment Training: Fundamentals course or the *Rational Environment User's Guide.*

This manual focuses on fundamental areas of the Environment necessary to begin work on small Ada® programs in single libraries. Some of the areas are: executing commands, managing windows, writing and debugging programs, and editing text files. Areas not included are multilibrary development, sophisticated use Rational Subsystems™, and optional products such as the Rational Design Facility, Rational Mail Utility, host-target development products, and so on.

RATIONAL

Chapter 1. Logging In and Out

Logging In

Begin with the terminal turned on.

1. Start the login sequence: Return
2. At the `Enter user name:` prompt, enter your username and press Return
3. At the `Enter password:` prompt, enter your password (it will not be echoed) and press Return
4. At the `Enter session name:` prompt, enter a session name and press Return (just press Return for the default session named S_1).

The Environment momentarily displays a message indicating the last time you were logged in, the screen goes blank, and the Environment session appears on the screen. A Login procedure in your home library is executed if it exists and is in the coded state.

Logging Out

Begin in any window.

1. Create a Command window: Create Command
2. Enter `quit` and press Promote

If no uncommitted (unsaved) images exist and if no programs requesting interactive input are running, the command is displayed in reverse video; the screen goes blank and you are logged out.

If any images were left without saving or promoting, or if a program requesting interactive input is running, an error message is displayed in the Message window indicating that images were left with unsaved changes. You can save all changed images (see below) and terminate any such running programs. Otherwise, enter `quit (true)` and press Promote. This logs you off the Environment without saving any uncommitted images.

Saving Changes

Begin in any window.

Saving changes one image at a time

1. Go to the Window Directory: Window - Definition
2. Place the cursor on a line containing an asterisk (*) in the Mod column.
3. Select the Window Directory entry: Object - —
4. Save the selected image: Enter

   The Mod column is now blank.

   Note that running programs requesting input still have a * in the Mod column. These programs must be terminated by killing their jobs (see "Killing Any Job" in Chapter 15).
5. Continue saving the changes desired by repeating the steps above.

Saving changes in all images in a single operation

1. Go to the Window Directory: Window - Definition
2. Place the cursor on the top line of the image: Image - Begin Of
3. Save all changes: Enter

All images that have been changed now have a blank in the Mod column.

Note that running programs requesting input still have a * in the Mod column. These programs must be terminated by killing their jobs (see "Killing Any Job" in Chapter 15).

Chapter 2. Getting Help

Getting Help on Help

To determine the available help for the Environment:

1. Ask for help: Help on Help

The Environment displays the available help options in the Help window.

Getting Help on a Specific Item

To get help on an Ada item (for example, a command) in an Ada or a Command window

Begin in the window containing the Ada item.

1. Place the cursor on the item for which you want help.
2. Ask for help on the designated item: Help

If help is available for the command, it is displayed in the Help window.

To get help on a named topic, command name or name fragment, and so on

1. Ask for help: Prompt For - Help

   The Environment creates a Command window and displays the command

   ```
   What.Does(Name => "");
   ```

2. At the prompt, enter the topic, command name, or command name fragment for the area of interest and press Promote

If more than one command related to that topic exists, all the related 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 displayed in the Help window.

If only one command about that topic exists, information about that command, including a brief command description and a list of any keys bound to the command, is 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 Keys

To determine what commands are bound to a key or key combination:

1. Ask for help on a key: Help on Key

   The Environment displays the following prompt in the Message window:

   ```
   Press key to be described:
   ```

2. Press the key or key combination of interest.

The command name bound to the key or key combination is displayed in the Message window. Additional help about the command, if any exists, is also displayed in the Help window.

Displaying Ada Specifications

To go to the Ada specification for an item described in the Help window:

Begin in the Help window in the entry for the message of interest.

1. Place the cursor on the line in the Help window containing the text for the Ada code for the item.
2. Ask for the definition of the designated item: Definition

If there is an Ada spec for the item, it is displayed and highlighted in an Ada window.

Displaying the Help Window

Begin in any window.

1. Ask to go to the Help window: Help Window

The Help window is brought onto the screen and the cursor is placed in it. You can now scroll through the contents of the window to view the help messages that have been requested since you logged in.

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. Ask for help on the error: Explain

Additional messages about the error appear in the Message window if the Environment has any more information to give you.

Chapter 3. Executing Commands

Creating and Executing a Command Window Program

A Command window program can contain any arbitrarily sized Ada code—for example, one-line Environment commands, multiple-line test programs, or Ada main programs.

Begin in any window.

1. Create a Command window: Create Command
2. Enter the program, formatting frequently for multiple-line programs: Format
3. Semanticize for multiple-line programs: Semanticize

   The Environment marks the errors that exist. Press Explain for further information about any errors.
4. Correct any errors and semanticize again.
5. Execute the command program: Promote

Expanding a Command Window

Begin in the Command window you want to expand.

1. Enlarge the window: Window - !

The window expands by four lines.

Shrinking a Command Window

Begin in the Command window you want to shrink.

1. Shrink the window: Window - .

The window shrinks by four lines.

Getting Command Completion

Begin in a Command window.

1. Enter some fragment of the command.
   - You may supply only a command name or name fragement. 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 pressed the Format key and it has added a semicolon after the name or name fragment).
2. Complete the command and provide prompting for any parameters: Complete

   If the command fragment is ambiguous, the complete operation fails and the Environment displays the possibilities in another window. Enter the necessary characters to make the command unique and press Complete again.

Moving to the Next Prompt or Underline

Begin in the Command window.

1. Move to the next item (highlighted or underlined): Next Item

The cursor is now placed at the next item (to the right or below).

Moving to the Previous Prompt or Underline

Begin in the Command window.

1. Move to the previous item (highlighted or underlined): Previous Item

The cursor is now placed at the next item (to the left or above).

Turning Off a Prompt

Begin with the cursor on the prompt that is to be turned into text.

1. Turn off the prompt: Item Off

Reexecuting the Same Command

Begin in the Command window containing the command to be reexecuted.

1. Execute the command: Promote

Changing and Reexecuting a Command

Begin with the cursor on the command to be changed.

1. Turn the command from a prompt into text: Item Off
   The command text can now be edited.
2. Execute the changed command: Promote

Entering a New Command in the Same Command Window

Begin with the cursor on the old command prompt.

1. Type the new command over the old command.

The old command prompt disappears.

Clearing a Command Window of Unneeded Text

Begin in the Command window to be cleared.

1. Clear the Command window: Edit

Note that the unneeded text in the Command window has been replaced with a statement prompt allowing entry of new commands.

Going Back to Previous Commands

A history of commands and Ada programs entered into a Command window is maintained. You can access and execute any of the commands in this sequential history.

Begin in the Command window.

Redisplaying a previous command in the historical sequence (undoing)

1. Redisplay the previous command: Object - U

Redisplaying a later command in the historical sequence (redoing)

1. Redisplay the next command: Object - R

Getting the Parameters of a Command Bound to a Key

Begin in any window.

1. Create a Command window with the parameters for a command bound to a key: Prompt For - command key

Chapter 4. Managing Windows

Finding a Window Using the Window Directory

Begin in any window.

1. Display the Window Directory: [Window] - [Definition]

   The Window Directory is displayed in a new window.
2. Place the cursor on the line of the Window Directory entry that names the window at which you want to look.
3. Ask to view the object: [Definition]

The indicated object appears in the same frame as the Window Directory window (or in an empty frame if one exists).

Deleting Windows from the Window Directory

Begin in the Window Directory window.

1. Place the cursor on the line of the window to be deleted.
2. Select the line: [Object] - [—]
3. Delete the window: [Object] - [D]

The window is removed from the Window Directory. This releases the image.

Moving between Windows

Moving to the window above (with vertical wraparound)

1. Move to the window above: [Window] - [↑]

Moving to the window below (with vertical wraparound)

1. Move to the window below: [Window] - [↓]

Expanding a Window

Begin in the window you want to expand.

1. Enlarge the window: Window - I

The window expands by four lines.

Shrinking a Window

Begin in the window you want to shrink.

1. Shrink the window: Window - D

The window shrinks by four lines.

Expanding Current Window to Include Next Frame

Begin in the window you want to expand.

1. Join the windows: Window - J

The current window expands to the size of the current window plus the window below, replacing any window that might have been on the screen. The window returns to its normal size automatically when the next object is viewed.

Expanding Current Window to Include Previous Frame

Begin in the window you want to expand.

1. Join the windows: Window - Delete

The current window expands to the size of the current window plus the window above, replacing any window that might have been on the screen. The window returns to its normal size automatically when the next object is viewed.

Transposing Windows

You can switch the location of a window with that of the window above it (with vertical wraparound).

Begin in the lower window.

1. Transpose the windows: Window - T

The cursor appears in the new lower window. It is in the same position that it was in when that window was last viewed.

Realigning the Windows on the Screen

Begin in any window.

1. Return windows to their default configuration: Window - Format

Removing a Window

You can remove a window from your screen in one of three ways.

Removing a window temporarily

This command removes the window from the screen and leaves it available in the Window Directory.

1. Place the cursor in the window you want to remove.
2. Delete the window: Window - D

Releasing an image permanently and saving the changes

This command releases the image and removes the window after saving the image. The window is no longer available in the Window Directory.

1. Place the cursor in the window you want to release.
2. Release the image: Object - X

Releasing an image permanently without saving the changes

This command abandons the image and removes the window. The window is no longer available in the Window Directory. Unsaved changes are discarded.

1. Place the cursor in the window you want to release.
2. Abandon the image: Object - G

Locking a Window on the Screen

Begin in the window you want to lock.

1. Lock the window: Window - Promote

An *at* sign (@) appears in the window banner. The window is not removed unless you explicitly remove it or unlock it.

Unlocking a Window on the Screen

Begin in the window you want to unlock.

1. Unlock the window: [Window] - [Demote]

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

Scrolling the Image

Begin in the window containing the image to be scrolled.

Scrolling the image up

1. Scroll the image up: [Image] - [↑]

Scrolling the image down

1. Scroll the image down: [Image] - [↓]

Scrolling to the beginning of the image

1. Scroll to the beginning of the image: [Image] - [Begin Of]

Scrolling to the end of the image

1. Scroll to the end of the image: [Image] - [End Of]

Scrolling the current line to the top

1. Scroll the current line to the top: [Window] - [Begin Of]

Scrolling the current line to the bottom

1. Scroll the current line to the bottom: [Window] - [End Of]

Chapter 5. Traversing the Environment

Viewing a Library

Begin in the world or directory that contains the library.

1. Place the cursor on the line containing the library.
2. View the library: Definition

A window appears, displaying the full pathname of the library underlined and listing additional library objects, such as Ada units or files, if they exist.

Viewing an Object in a Library

Begin in the library containing the object.

1. Place the cursor on the line of the library object you want to view.
2. View the object: Definition

A window displaying the object appears.

Viewing a Library's Parent

Begin in the library.

1. View the parent: Enclosing

A window containing the parent library appears.

Viewing Your Home Library

Begin in any library.

1. View your home library: Home Library

A window containing your home library appears.

Viewing the Specification of an Environment Package

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

Begin in any window.

1. Get a prompt for the Definition command: Prompt For - Definition
2. Enter the simple name of the Ada unit at the prompt for the Name parameter preceded by the \ character (for example, "\Compilation").
3. Execute the command: Promote

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

Chapter 6. Using General Editing Operations

Selecting an Arbitrary Region of Text

Begin in the window containing the text to be selected.

1. Move the cursor to the start of the region of text to be selected.
2. Define the start of the region: Region - [
3. Move the cursor to the end of the region of text.
4. Define the end of the region: Region -]

The selected region is highlighted.

Moving Selected Text

Begin in the window containing the text to be moved.

1. Select the region of text.
2. Move the cursor to the location in which the text will be moved. You can move text within the same image or to some other image.
3. Move the region of text: Region - M

The highlighted region of text is deleted from its original location and appears in the new location.

Copying Selected Text

Begin in the window containing the text to be copied.

1. Select the region of text.
2. Move the cursor to the location in which the text will be copied. You can copy text within the same image or into some other image.
3. Copy the region of text: Region - C

The region of text appears in its original location and in the new location.

Searching for a String

Begin in the text in which you want to search for the string.

1. Move to the beginning of the image: [Image] - [Begin Of]
2. Start the search command (enter composing mode): [Control][S]
3. Enter the target string, without quotes. Note that the characters you type in composing mode appear at the SEARCH prompt in the Message window.
4. Start the actual search (enter search mode): [Control][S]

   If the target string is found, the cursor is positioned one character after the target string.
5. To get to each additional occurrence of the string: [Control][S]
6. To return to a previous occurrence of the string: [Control][R]
7. To cancel the search, press any key—for example, [↓].

   The SEARCH prompt is removed from the Message window.

Searching and Replacing a String

Begin in the text with the string to be changed.

1. Move to the beginning of the image: [Image] - [Begin Of]
2. Start the search/replace command: [Meta][S]
3. At the SEARCH prompt in the Message window, enter the target string, without quotes.
4. Press [Next Item] to move to the REPLACE prompt.
5. At the REPLACE prompt in the Message window, enter the replacement string, without quotes.
6. Start the actual search/replace: [Meta][S]

   The Environment places the cursor one character after the target string.
7. To replace the target string: [Meta][S]

   The Environment replaces the string and places the cursor one character after the next occurrence of the target string.
8. To get to each additional occurrence of the string without changing the string: [Control][S]
9. To replace a previous occurrence of the string: [Meta][R]
10. To abort searching and replacing, press any key—for example, [↓].

    The SEARCH and REPLACE prompts are removed from the Message window.

Searching and Replacing All Occurrences of a String

Begin in the text with the string to be changed.

1. Move to the beginning of the image: [Image] - [Begin Of]
2. Start the search/replace command: [Meta][S]
3. At the SEARCH prompt in the Message window, enter the existing string, without quotes.
4. At the REPLACE prompt in the Message window, enter the new string, without quotes.
5. Start the actual search and global replace: [numeric -] - [numeric 1] - [Meta][S]

   (Use the numeric keypad to enter the −1.)

   The Environment replaces all occurrences of the target string and displays the number of occurrences in the Message window.

Deleting Text

Text such as characters, words, lines, and regions can be deleted. Text can be deleted from varying cursor positions.

- Delete the character at the cursor: [Control][D]
- Delete the character before the cursor position (backspacing): [Delete]
- Delete the entire word: [Word] - [D]
- Delete from the cursor to the end of the word: [Word] - [K]
- Delete from the cursor to the beginning of the word: [Word] - [Delete]
- Delete the entire line: [Line] - [D]
- Delete from the cursor to the end of the line: [Line] - [K]
- Delete from the cursor to the beginning of the line: [Line] - [Delete]
- Delete the selected text: [Region] - [D]

Joining Lines

This command joins the line on which the cursor is located with the following line.

1. Move the cursor to any position on the first line of the two lines to be joined.
2. Join the second line to the end of the first line: [Line] - [J]

Transposing Text

Transposing characters

This command switches the character that the cursor is on with the previous character. Assume, for example, that character 2 follows character 1, and you want character 1 to follow character 2.

1. Move the cursor to character 2.
2. Transpose the character that the cursor is on and the previous character: Control T

Transposing words

This command switches the word that the cursor is on with the previous word. Assume, for example, that word 2 follows word 1, and you want word 1 to follow word 2. Word terminators are blanks, underscores, semicolons, or periods.

1. Move the cursor to any place on word 2.
2. Transpose the word that the cursor is on and the previous word: Word - T

Transposing lines

This command switches the line that the cursor is on with the previous line. Assume, for example, that line 2 follows line 1, and you want line 1 to follow line 2.

1. Move the cursor to any place on line 2.
2. Transpose the line that the cursor is on and the previous line: Line - T

Changing the Case of Text

The case of text such as characters, words, lines, and regions can be changed to lowercase, uppercase, or initial capitals. Begin with the cursor anywhere in the text to be changed.

- Capitalize a character: Control >
- Lowercase a character: Control <

- Uppercase a word: Word - >
- Lowercase a word: Word - <
- Capitalize a word: Word - ^

- Uppercase a line: Line - >
- Lowercase a line: Line - <
- Capitalize a line: Line - ^

- Uppercase a selected region: Region - >
- Lowercase a selected region: Region - <
- Capitalize a selected region: Region - ^

Chapter 7. Writing Text Files

Creating a File

Begin in the library in which you want the file.

1. Create a file: Create Text

   A Command window with the Text.Create command and its parameter is created.

2. At the Image_Name prompt, enter the name of the file to be created and press Promote

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

Viewing a File

Begin in the library containing the file.

1. Move the cursor to the line containing the file declaration.
2. Go to the definition: Definition

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

Editing an Existing File

Begin in the library containing the file.

1. Move the cursor to the line containing the file declaration.
2. Select the file to be edited: Object - –
3. Edit the selected file: Edit

   The Environment displays the image of the object in a window. You are now ready to edit the file.

4. Save the image periodically by pressing Enter
5. When you have finished editing, promote the file to a read-only image by pressing Promote

Saving a File

A file can be saved in one of two ways.

Saving a file (close for editing)

When you have made some changes and you want to save them and terminate editing:

1. Place the cursor in the window that has the image of the file.
2. Promote the image to a read-only image: Promote

This command saves the image of the file and allows others to access it.

Saving a file (leave open for editing)

When you have made some changes and you want to save them but continue editing:

1. Place the cursor in the window that has the image of the file.
2. Commit the image: Enter

This command saves the image of the file, and you retain update access.

Setting Tabs

Begin in the text.

1. Create a Command window.
2. To set tab stops at every nth column, enter `set.tab_width(n)` and press Promote

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

Setting Overwrite Mode On

Begin in the text.

1. Set overwrite mode on: Image - O

The banner is updated to indicate that overwrite mode is in effect in this window.

Setting Insert Mode On

Begin in the text.

1. Set insert mode on: Image - I

Setting Wordwrap for Text

Begin in the text.

1. Turn fill mode on: Image - F

The banner shows that fill mode is in effect and indicates the column number. The column number default is 72.

Changing the Wordwrap Column

Begin in the text.

1. Create a Command window.
2. To set a different wordwrap column, enter `set.fill_column` and press Complete
3. At the prompt, enter `n`, where `n` is the desired column number, and press Promote

Turning Wordwrap Off

Begin in the text.

1. Turn fill mode off: Image - X

The banner is updated to remove the fill mode indicator and fill column number.

RATIONAL

Chapter 8. Writing Ada Programs

Libraries are of two kinds: directories and worlds. Programs can be written in either kind of library.

Creating an Ada Package Specification

Begin in the library that will contain the Ada unit.

1. Create a workspace: Create Ada

   A new window is created with a `comp_unit` prompt for you to begin editing.

2. Enter the contents of the specification in the new window at the `comp_unit` prompt.

   Use Create Private for building the private part of the specification, if appropriate.

3. Format frequently by pressing Format

   The Environment marks any errors that exist. Use Explain for information about any errors.

4. Semanticize frequently by pressing Semanticise

   The Environment marks any errors that exist. Use Explain for information about any errors.

   The first time you semanticize, a temporary name appears in the banner of the Ada unit you are editing and in the library that contains the Ada unit. A temporary name is of the form `_Ada_#_`, where # is some number.

5. Correct any errors.

6. Promote the specification to the installed state: Promote

The Environment replaces the temporary name in the library with the Ada name for the unit specification.

Creating an Ada Package Body

Begin in the package specification.

1. Use [Create Body] to build the skeletal package body.

   A new window appears with the skeletal package body for you to edit.
2. Enter the contents of the body.
3. Format and semanticize frequently.

   The Environment marks any errors that exist. Use [Explain] for information about any errors.

   The first time you semanticize, a temporary name appears in the banner of the Ada unit you are editing and in the library that contains the Ada unit. A temporary name is of the form `_Ada_#_`, where # is some number.
4. Correct any errors.
5. Promote the body to the installed state: [Promote]

The Environment replaces the temporary name in the library with the Ada name for the unit specification.

Creating an Ada Subprogram

Begin in the library that is to contain the Ada unit.

1. Create a workspace: Create Ada

   The Environment creates a new window with a `comp_unit` prompt.
2. Enter the body of the subprogram.
3. Format and semanticize the unit.

   The Environment marks any errors that exist. Use Explain for information about any errors.

   The first time you semanticize, a temporary name appears in the banner of the Ada unit you are editing and in the library that contains the Ada unit. A temporary name is of the form `_Ada_#_`, where # is some number.
4. Correct any errors.
5. Promote the subprogram to the installed state: Promote

The Environment replaces the temporary name in the library with the Ada name for the unit. It also creates a separate specification for the unit in the library.

Creating a Subunit

Begin in the Ada unit that will contain the subunit.

1. Enter the Ada subunit stub notation. You might enter, for example, `procedure foo is separate;`
2. Format.
3. Place the cursor on the stub.
4. Select the stub: Object - ←
5. Edit the selected stub: Edit

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

Importing Units

To import units, see "Adding Links—Simple Method" in Chapter 12.

Adding a Statement, Declaration, or Comment

Adding to an Ada unit in the source state

Begin in the Ada unit in which you want to make the addition.

1. Edit the Ada unit, if it is still in read-only mode: Edit
2. Go the position where the new statement, declaration, or comment is to be added.
3. Enter the changes.
4. Format and semanticize.
5. Correct any errors.

Adding to an Ada unit in the installed or coded state

Begin in the Ada unit in which you want to make the addition.

1. If the Ada unit is a package specification or if the addition you want to make contains only Ada comments, skip to the next step.

   If it is already coded, demote the Ada unit to the installed state: Install Unit
2. Go to the position where the new statement, declaration, or comment is to be added.
3. Open an insertion point: Object - I

   A new window appears with the banner labeled either `statement` or `declaration`, depending on the location of the insertion point.

   The library now contains a temporary name of the form `_Ada_#_`, where # is some number, under the library unit you are editing.
4. Enter the new statement, declaration, or comment.

   Note that multiple statements, declarations, or comments can be entered per insertion point.
5. Format and semanticize.
6. Correct any errors.
7. Promote the statement, declaration, or comment: Promote

The new window disappears, and the prompt in the unit is replaced by the actual statement, declaration, or comment. The temporary name in the library is removed.

Changing a Statement, Declaration, or Comment

Making changes in an Ada unit in the source state

Begin in the Ada unit in which you want to make the change.

1. Edit the Ada unit, if it is still in read-only mode: Edit
2. Go to the position where the statement, declaration, or comment is to be changed.
3. Enter the changes.
4. Format and semanticize.
5. Correct any errors.

Making changes in an Ada unit in the installed or coded state

Begin in the Ada unit in which you want to make the change.

1. If the Ada unit is a package specification or if the change you want to make consists only of Ada comments, skip to the next step.

   If it is already coded, demote the unit to the installed state: Install Unit
2. Go to the end of the statement, declaration, or comment to be changed.
3. Select the entire statement, declaration, or comment: Object - ←
4. Edit the selected statement, declaration, or comment: Edit

   The selected statement, declaration, or comment becomes a prompt, and a window with the statement, declaration, or comment appears on the screen.

   The library now contains a temporary name of the form `_Ada_#_`, where # is some number, under the library unit you are editing.

   Note that if the selected declaration has dependents, the edit operation will not succeed until all dependents are demoted to source.
5. Enter the changes.

   Note that multiple declarations, statements, or comments can be entered.
6. Format and semanticize.
7. Correct any errors.
8. Promote the statement, declaration, or comment: Promote

The new window disappears, and the prompt in the unit is replaced by the actual statement, declaration, or comment. The temporary name in the library is removed.

Deleting a Statement, Declaration, or Comment

Deleting in an Ada unit in the source state

Begin in the Ada unit in which you want to make the change.

1. Edit the Ada unit, if it is still in read-only mode: Edit
2. Go the position where the statement, declaration, or comment is to be deleted.
3. Use line delete or region delete to remove the statement, declaration, comment.

The unit remains in the source state for further editing.

Deleting in an Ada unit in the installed or coded state

Begin in the Ada unit in which you want to make the change.

1. If the Ada unit is a package specification or if the deletion you want to make contains only Ada comments, skip to the next step.

   If it is already coded, demote the unit to the installed state: Install Unit
2. Go to the end of the statement, declaration, or comment to be deleted.
3. Select the entire statement, declaration, or comment: Object - –
4. Delete the selected statement, declaration, or comment: Object - D

The selected statement, declaration, or comment is removed.

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

Changing the Name or Kind of an Ada Unit

Changing the name or kind of an Ada unit in the source state

Begin in the library containing the Ada unit to be changed.

1. Move the cursor to the line containing the Ada unit.
2. Select the Ada unit: Object - —
3. Edit and withdraw the selection: Withdraw Unit

   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.

The temporary name in the library is replaced by the new actual name for the Ada unit when you promote the unit. The unit is still in the source state to allow continued editing.

Changing the name or kind of an Ada unit in the installed or coded state

Begin in the library containing the Ada unit to be changed.

1. Move the cursor to the line containing the Ada unit.
2. Select the Ada unit: Object - —
3. Edit and withdraw the selection: Withdraw Unit

   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. Enter the changes.
5. Format and semanticize.
6. Correct any errors.
7. Promote the unit: Promote

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

Adding a Subprogram to a Package

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

Adding to an Ada unit in the source state

Begin in the package specification in which you want to add the subprogram specification.

1. Edit the Ada unit, if it is still in read-only mode: Edit
2. Go to the position in the package where the new subprogram specification is to be added.
3. Enter the new subprogram specification.
4. Format and semanticize.
5. Correct any errors.
6. Select the subprogram specification: Object - —
7. Create the body: Create Body

   The skeletal subprogram body is placed at the end of the existing package body.
8. Enter the subprogram body.
9. Format and semanticize frequently.
10. Correct any errors.

Adding to an Ada unit in the installed or coded state

Begin in the package specification in which you want to add the subprogram specification.

1. Go to the position in the package where the new subprogram specification is to be added.
2. Open an insertion point: Object - I

   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. Format and semanticize.
5. Correct any errors.
6. Promote the declaration: Promote

   The new window disappears and the prompt in the package specification is replaced with the added subprogram specification. The temporary name in the library disappears.
7. Select the subprogram specification: Object - ←
8. Create the body: Create Body

   A new window appears on the screen with the skeletal subprogram body.
9. Enter the subprogram.
10. Format and semanticize frequently.
11. Promote the subprogram body: Promote

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

Making a Package Body or Subprogram Body into a Subunit

Begin in the parent unit containing the declaration stub in either the source or the installed state.

1. Select the unit that you want to make into a subunit: Object - ←
2. Make the selected unit separate: 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

Begin in the parent Ada unit in either the source or the installed state.

1. Select the subunit stub.
2. Make the subunit in-line: Make Inline

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

Demoting a Unit and Its Dependents

Begin in the library that contains the program unit.

1. Place the cursor on the line containing the program unit to be demoted.
2. Select the unit to be demoted: Object - ←
3. Demote the program unit: Source (This World)

The progress of the command is displayed in the Environment I/O window. The unit, plus any units that depend on it, is demoted to source.

Making a Library Program Executable

Begin in the library that contains the program.

1. Make the program executable: Code (This World)

All units in the library are promoted to the coded state. The progress of the command is displayed in the Environment I/O window.

Executing a Library Program

Begin in the library containing the program.

1. Create a Command window.
2. Enter the Ada name for the program.
3. Execute the program: [Promote]

The Environment then executes the program just as it executes any Environment command.

Saving the Changes of Incomplete Units

Begin in the Ada unit that is incomplete—that is, the unit still may have errors or you want to do further development on the unit before promoting it.

1. Save the image: [Enter]

A message appears in the Message window indicating that the unit has been saved *(committed)*. The banner of the Ada unit now has a blank in the first character position.

Setting Overwrite Mode On

Begin in the Ada unit you are editing.

1. Set overwrite mode on: [Image] - [O]

The banner is updated to indicate that overwrite mode is in effect in this window. Overwrite mode is set on a window-by-window basis.

Setting Insert Mode On

Insert mode is the default. Begin in the window that is currently in overwrite mode.

1. Set insert mode on: [Image] - [I]

The banner is updated to remove the overwrite mode indicator.

Chapter 9. Browsing Ada Programs

Getting the Definition or Use of an Identifier

Begin with the cursor on the identifier.

1. Select the identifier: [Object] - [—]
2. Go to the definition: [Definition]

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 for Ada units provided as part of the Environment (for example, for viewing the specification for package Compilation, which contains the compilation commands).

Begin in any window.

1. Get a prompt for the Definition command: [Prompt For] - [Definition]
2. Enter the simple name of the Ada unit at the prompt for the Name parameter preceded by the \ character (for example, "\Compilation").
3. Execute the command: [Promote]

Note that 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.

Viewing a Unit's Specification from Its Body

Begin in the body.

1. Go to the specification: [Other Part]

A window containing the specification appears.

Viewing a Unit's Body from Its Specification

Begin in the specification.

1. Go to the body: [Other Part]

A window containing the body appears.

Viewing a Unit's Parent

Begin in the unit.

1. Go to the parent: [Enclosing]

A window containing the parent object appears.

Showing the Using Occurrences of a Defined Ada Name

Begin in the window containing the Ada name of interest.

1. Place the cursor on an occurrence of the Ada name.
2. Select the Ada name of interest: [Object] - [←]
3. View the using occurrences: [Show Usage]
4. The using occurrences of the Ada name within the current unit are underlined. Use [Next Item] or [Previous Item] to step through.

   For using occurrences of the Ada name in other units, a window containing the names of these units appears.
5. Place the cursor on a unit.
6. Select the unit: [Object] - [←]
7. View the unit with the using occurrence: [Definition]

   A window appears displaying the selected unit with all occurrences of the Ada name of interest underlined.
8. Use [Next Item] or [Previous Item] to step through.

Chapter 10. Debugging

Starting the Debugger

Begin in the Command window containing the name of the program to be debugged.

1. Invoke the Debugger: Meta Promote

The Debugger window appears, and a debugging session begins.

Program execution does not begin until further debugging commands are entered.

Stopping the Debugger

A debugging session is terminated automatically when you begin to debug a new job or when you log off.

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)*.

Displaying the Value of a Program Variable

Begin in any window.

1. Place the cursor on an occurrence of the program variable.
2. Select the program variable: Object - —
3. Display the value: Put

The value of the variable is displayed in the Debugger window.

Displaying the Call Stack

Begin in any window.

1. Display the stack: [Stack]

The call stack is displayed in the Debugger window with the most current call on the top of the stack (it is frame number one: "_1").

Displaying Source for a Call Stack Frame

Begin in the Debugger window.

1. Display the stack: [Stack]
2. Place the cursor on the frame you want to display.
3. Select the frame: [Object] - [←]
4. Display the source for the frame: [Show Source]

The Ada unit corresponding to the frame is displayed with the program counter location (either current or saved) highlighted.

Displaying Parameters for a Call Stack Frame

Begin in the Debugger window.

1. Display the stack: [Stack]
2. Place the cursor on the frame for which you want to display the parameters.
3. Select the frame: [Object] - [←]
4. Display the parameters for the frame: [Put]

Stepping Through the Program

You can step in one of two ways. Note that in either case you can step multiple times with a single command by pressing a numeric prefix key ([numeric n]) before you press the key to step.

Begin in any window.

Stepping by every statement

1. Press [Run]

Stepping by statements without stopping in called subprograms

1. Press [Run Local]

Executing the Program

Begin in any window.

1. Execute the program: [Execute]

The program runs to completion or until an exception or breakpoint is encountered.

Setting Up Exception Handling

The Debugger stops when any exception is encountered, unless that exception has been propagated.

Begin in an Ada window containing the unit that declares the exception or a unit that handles the exception.

Propagating a particular exception

1. Place the cursor on an occurrence of the exception name.
2. Select the exception: [Object] - [←]
3. Press [Propagate]

Catching a previously propagated exception

1. Place the cursor on an occurrence of the exception name.
2. Select the exception: [Object] - [←]
3. Press [Catch]

Setting Breakpoints

Begin in the window displaying the Ada unit in which you want to set a breakpoint.

1. Place the cursor on the statement or declaration in the Ada unit.
2. Select the entire statement or declaration by pressing [Object] - [←] repeatedly.
3. Set the breakpoint: [Break]

A breakpoint number is assigned. This breakpoint is in effect until the Debugger session terminates or until it is explicitly deactivated.

Showing Breakpoints

Begin in any window.

1. Show breakpoints: [Show Breaks]

The display shows all active and inactive breakpoints.

Removing Breakpoints

Begin in any window.

Removing all breakpoints

1. Remove all breakpoints: Remove Breaks

Removing a specific breakpoint

1. Prompt for the remove command: Prompt For - Remove Breaks
2. At the Breakpoint prompt, enter the number of the breakpoint you want deactivated and press Promote

Modifying a Program Variable

Begin in the window displaying the program variable.

1. Place the cursor on an occurrence of the program variable you want to change.
2. Select the program variable: Object —
3. Prompt for the modify command: Modify
4. At the New_Value prompt, enter the desired new variable value (in double quotes) and press Promote

Returning to the Point of Program Suspension

Begin in any window.

1. Go to the program suspension point: Show Source

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

Displaying the Debugger Window

Begin in any window.

1. Display the Debugger window: Debugger Window

The Debugger window appears on the screen and the cursor is in it.

Chapter 11. Managing Libraries

Controlling the Library Display

Begin with the cursor in the library.

Toggling information on library objects

1. Move to the beginning of the library: [Image] - [Begin Of]
2. Change the display: [Explain]

Repeating this command toggles the library display so that you view one of the following: only the names of the library objects; the name and the type of library objects; and the name, type, Ada unit state plus update information.

Showing more detail on the objects in the library

1. Show more detail: [Object]- [!]

This causes deleted units, versions, and so on to be added to the library display.

This step can be repeated if necessary until the desired detail level is reached.

Showing less detail on the objects in the library

1. Show less detail: [Object]- [.]

This causes deleted units, versions, and so on to be removed from the library display.

This step can be repeated if necessary until the desired detail level is reached.

Creating Libraries

Creating a directory

Begin in the directory or world that is to contain the new directory.

1. Create the directory: Create Directory

   The Environment creates a Command window containing the Library.Create_Directory command and prompts for its parameters.
2. At the Name prompt, enter the name for the new directory and press Promote

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

Creating a world

Begin in the directory or world that is to contain the new world.

1. Create the world: Create World

   The Environment creates a Command window containing the Library.Create_World command and prompts for its parameters.
2. At the Name prompt, enter the name for the new world and press Promote

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

By default, this world has links to commonly used Ada and Environment packages such as Text_Io, Calendar, and String_Tools. These links are from the model world !Model.R1000.

Deleting Objects in a Library

Deleting a library

Begin in the library containing the library to be deleted.

1. Place the cursor on the line containing the library to be deleted.
2. Select the library to be deleted: Object - ←
3. Create a Command window: Create Command
4. Enter `compilation.delete` and press Promote

The I/O window displays the progress and results of the Delete command. When the command is complete, the library to be deleted disappears from the library.

Deleting an Ada unit or file

Begin in the library containing the object to be deleted.

1. Place the cursor on the line containing the object to be deleted.
2. Select the object: Object - ←
3. Delete the object: Object - D

If an Ada unit has no dependents, the declaration is removed from the library.

Undeleting Objects or Previous Versions in a Library

Begin in the library containing the deleted object or version.

1. Expand detail in the library (if necessary) so you can see the object or version to be undeleted: Object - ↓

   Repeat as necessary until you can see the deleted object or version you want to undelete. A deleted object is enclosed in braces ({}) to indicate that it is deleted. A previous version has its name prefixed with a minus (–), indicating that it is not the default version.
2. Select the object or version to undelete: Object - ←
3. Undelete it: Object - U

The object or version is now undeleted and is displayed without the braces around it or without the minus in front of it.

Copying Objects in a Library

Copying into a different library

Begin in the library containing the object (library, Ada unit, file) to be copied.

1. Place the cursor on the object to be copied.
2. Select the object to be copied: Object - ←
3. Place the cursor in the new library to which the existing object is to be copied.
4. Copy the selected object: Object - C

   A Command window appears with the Library.Copy command and prompts for its parameters. The parameter names are supplied automatically by the Environment.
5. Press Promote

The progress of the command is displayed in the Environment I/O window.

Copying into the same library

Begin in the library containing the object (library, Ada unit, file) to be copied.

1. Select the object to be copied: [Object] - [—]
2. Copy the selected object: [Object] - [C]

   A Command window appears with the Library.Copy command and prompts for its parameters.
3. At the To prompt, enter the name of the object into which you want to copy.
4. Press [Promote]

The progress of the command is displayed in the Environment I/O window.

Moving Objects in a Library

Moving to a different library

Begin in the library containing the object (library, Ada unit, file) to be moved.

1. Place the cursor on the object to be moved.
2. Select the object to be moved: [Object] - [—]
3. Place the cursor in the new library to which the existing object is to be moved.
4. Move the selected object: [Object] - [M]

   A Command window appears with the Library.Move command and prompts for its parameters. The parameter names are supplied automatically by the Environment.
5. Press [Promote]

The progress of the command is displayed in the Environment I/O window.

Moving to the same library

This is equivalent to renaming a library object. See "Renaming Objects in a Library," below.

Renaming Objects in a Library

Begin in the library structure containing the object (library, Ada unit, file) to be renamed.

1. Select the object to be renamed: [Object] - [—]
2. Create a Command window: [Create Command]
3. Enter `library.rename` and press [Complete]
4. At the To prompt, enter the new name and press [Promote]

The progress of the command is displayed in the Environment I/O window. Ada units are demoted to source.

Printing Objects Contained in a Library

Printing a file or an Ada unit

Begin in the library containing the object to be printed.

1. Move the cursor to the line containing the object to be printed.
2. Select the object: Object - —
3. Print the object: Print

The progress and status are displayed in the Message window. A listing appears on the printer.

Printing a library, its units, and its subunits

Begin in the library containing the objects to be printed.

1. Print: Prompt For - Print

   A Command window appears with the `Queue.Print` command and prompts for its parameters.
2. At the Name prompt, enter the wildcard symbol ? and press Promote

The progress and status are displayed in the Message window. A listing appears on the printer.

Chapter 12. Managing Links

Listing Links—Simple Method

Begin in the world for which you want to see the links.

1. Create a Command window: Create Command
2. Enter `links.display` and press Promote

A list of the links appears in the standard I/O window.

Adding Links—Simple Method

Begin with the cursor in the world to which you want to add the link.

1. Create a Command window: Create Command
2. Enter `links.add` and press Complete
3. At the `Source` prompt, enter the full pathname of the Ada unit to which you want the link to refer and press Promote

The new link is added to the world. The link name is the simple Ada name derived from the full pathname.

Getting the Pathname for an Environment Package

Begin in any window.

1. Create a command window: Create Command
2. Enter `library.resolve` and press Complete
3. At the `Name_Of` prompt, enter the simple name of the Ada unit for which you want the pathname prefixed with the \ character (for example, `\Text_Io`).
4. Execute the command: Promote

The full pathname is displayed in the I/O window. If you want to use this pathname as a parameter to another command, you can select the text of the pathname in the I/O window and then copy this region into a Command window.

Note that this shortcut for getting pathnames works for most Environment packages. If the shortcut fails, an error message appears, and you have to look for the pathname in the World ! section of the Reference Summary (in Volume 1 of the *Rational Environment Reference Manual*) or in the reference manual for the product area in question.

Editing Links for a World

Begin in the world for which you want to edit the links.

1. Create a Command window: Create Command
2. Enter `links.edit` and press Promote

A window displaying the links appears. You can now edit the links. See the individual editing operations that follow.

Controlling the Link Display

Begin with the cursor in the link display.

Toggling the order of the link display

1. Change display order: Object - !

Repeating this command toggles the display so that it appears alphabetically either by source name or by link name.

Toggling the contents of the link display

1. Change display contents: Object - .

Repeating this command toggles the display so that you view one of the following: only internal links, only external links, or all links.

Inserting a New Link

Begin with the cursor in the link display.

1. Open an insertion point: Object - I

   A Command window appears attached to the link display window with the Insert command and its parameter.
2. At the prompt, enter the full pathname of the Ada unit to which you want the link to refer and press Promote

The link display is updated to show the new link. The link name is the simple Ada name derived from the full pathname.

Deleting a Link

Begin with the cursor in the link display.

1. Move to the link you want to delete.
2. Select that link: [Object] - [–]
3. Delete the link: [Object] - [D]

The link is deleted and the link display is updated.

Viewing the Source of a Link

Begin with the cursor in the link display.

1. Move to the link whose source you want to view.
2. Select that link: [Object] - [–]
3. Go to the definition: [Definition]

A window appears containing the definition of the Ada unit to which the link refers.

Exiting from the Link Display

Begin with the cursor in the link display.

1. Release the link image: [Object] - [X]

The window containing the link display disappears.

Adding a Set of Links

Begin in the world to which you want to add a set of links.

1. Create a Command window: [Create Command]
2. Enter `links.add` and press [Complete]
3. At the `Source` prompt, enter a name (using substitution characters and wildcards, if desired) that specifies the complete set of links and press [Promote]

All links are added.

Replacing a Link

Begin in the world containing the link you want to replace.

1. Create a Command window: [Create Command]
2. Enter `links.replace` and press [Complete]
3. At the `Source` prompt, enter the new source name you want to have associated with an existing link and press [Promote]

The source for the link is replaced.

RATIONAL

Chapter 13. Managing Session Switches

Editing Session Switches

Begin in any window.

1. Create a Command window: [Create Command]
2. Enter `switches.edit_session_attributes` and press [Promote]

A window displaying the session switches appears. You can now edit the switches. A session switch file called *Current_Session_Name*_`Switches` appears in your home library, if it does not already exist.

Controlling the Session Switch Display

Begin with the cursor in the session switch display. Two commands toggle the session switches display so that you see one of the following views: all switches or nondefault switches (switches that you have modified).

1. Change the display to all switches: [Object] - [!]
2. Change the display to nondefault switches: [Object] - [.]

Modifying Session Switch Values

Begin with the cursor in the session switch display.

Modifying a Boolean switch

1. Place the cursor on the session switch whose value you want to modify.
2. Edit the selected session switch: Edit

   The value toggles between true and false. The session switch display is updated to show the new value.
3. Save the session switch image: Enter

Session switches take effect at varying times: immediately, at login, or when next displaying the object image.

Modifying a non-Boolean switch

1. Place the cursor on the session switch whose value you want to modify.
2. Edit the selected session switch: Edit

   A Command window appears with the Change command and a prompt for its parameter.
3. At the prompt, enter the new parameter value and press Promote

   The session switch display is updated to show the new value.
4. Save the session switch image: Enter

Session switches take effect at varying times: immediately, at login, or when next displaying the object image.

Getting Help on Session Switches

Begin with the cursor in the session switch display.

Getting an explanation

1. Place the cursor on the session switch for which you want to have further information.
2. Ask for help: Explain

An explanation of the session switch, if it exists, appears in the switch display below the selected session switch.

Removing an explanation

1. Place the cursor on the explanation that you want to remove.
2. Remove the explanation: Explain

The explanation disappears from the session switch display.

Saving Session Switches

Begin with the cursor in the session switch display.

1. Save the image: Enter

A message appears in the Message window indicating that the session switches have been saved *(committed)*.

Exiting from the Session Switch Display

Begin with the cursor in the session switch display.

1. Release the switch image: Object - X

The window containing the session switch display disappears.

Chapter 14. Managing Searchlists

Editing the Searchlist for a Session

Begin in any Command window.

1. Enter `search_list.show_list` and press Promote

A window displaying the session searchlist appears. You can now edit your searchlist.

Adding a Component to a Searchlist

Begin with the cursor in the searchlist display.

1. Move to the line where the new entry is to be added.
2. Open an insertion point: Object - I

   A Command window appears with the Add command and prompts for its parameters.
3. At the `Component` prompt, enter the new searchlist entry and press Promote

The searchlist display is updated to show the new entry.

Deleting a Component from a Searchlist

Begin with the cursor in the searchlist display.

1. Put the cursor on the searchlist component you want to delete.
2. Select the searchlist component: Object - –
3. Delete the searchlist component: Object - D

The entry is deleted and the display is updated.

Replacing One Component with Another

Begin with the cursor in the searchlist display.

1. Select the entry to be replaced: Object - —
2. Create a Command window: Create Command
3. Enter `replace` and press Complete
4. At the `New_Component` prompt, enter the new entry and press Promote

The old entry is replaced with the new one.

Viewing the Library Named by a Searchlist Entry

Begin with the cursor in the searchlist display.

1. Move to the searchlist entry you want to view.
2. Go to the definition: Definition

A window appears containing the library.

Exiting from the Searchlist Display

Begin with the cursor in the searchlist display.

1. Release the searchlist image: Object - X

The window containing the searchlist disappears.

Chapter 15. Managing Jobs

Disconnecting from a Job

1. Disconnect the job: 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 that logging out does not terminate disconnected jobs that are still executing unless these jobs attempt to perform input or output to Editor windows.

Reconnecting to a Job

Begin in any window.

1. Determine the number of the job to be reconnected. The job number is displayed on the banner of the I/O window for the job (if used). Otherwise, to display all the jobs currently running on the system, press What Users
2. Get a prompt for the connect command: Prompt For - Job Connect
3. At the The_Job parameter, enter the number of the job and press Promote

Killing the Current Job or the Last Job Created

Begin in any window.

1. Kill the last job: Job Kill

A job-abort message is displayed in the Message window.

Killing Any Job

Begin in any window.

1. Disconnect from the current job if necessary: Control G
2. Determine the number of the job to be killed. The job number is displayed on the banner of the I/O window for the job (if used). Otherwise, to display all the jobs currently running on the system, press What Users
3. Prompt for the command to kill the job: Prompt For - Job Kill
4. Enter the job number at the The_Job prompt and press Promote

   Note that the default job number is that of the job from which you just disconnected.

A message is displayed in the Message window indicating that the job has been killed.

Chapter 16. Customizing Your Workspace

Building Macros

You can bind a series of keystrokes to a single key by building a macro.

Defining the macro

1. Start the definition: Mark - Begin Of
2. Press the keystrokes that are to make up the macro.
3. End the definition: Mark - End Of

Executing the macro

1. Execute the last macro you entered: Mark - Promote

Binding the macro to a function key

1. Press Mark - Definition

   You are prompted in the Message window for a key to bind to the last macro entered.
2. Press the key to be bound.

The key remains bound only until you log out, unless you explicitly save it.

Saving the macro

1. Create a Command window.
2. Enter `macro.save` and press Promote

All macros currently bound to keys are permanently saved.

Defining Your Own Login Procedure

Begin in your home library.

1. Create a procedure named Login with the commands you want to have executed each time you log into the Environment.

   See "Creating an Ada Subprogram" in Chapter 8 for details.
2. Promote the procedure to the coded state: Code Unit

The Login procedure is now executed automatically as part of the login process.

Rebinding Keys

Before starting, you may want to press Help on Key to see if the key is already bound.

You can rebind commands to keys in one of two ways.

Begin in any window.

Rebinding temporarily

1. Create a Command window: Create Command
2. Enter `key.define` and press Complete
3. 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, press 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.
4. At the Command_Name prompt, enter the name of the command you want bound to this key and press Promote

The new key binding is in effect until you log out.

Rebinding permanently

Begin in your home world.

1. Create a procedure named Rational_Commands by copying the text from the template in !Machine.Editor_Data.Rational_User_Commands into an Ada window.

   See "Creating an Ada Subprogram" in Chapter 8 for details.
2. Edit the body of Rational_Commands so that the case statement contains alternatives for those keys you want to rebind.
3. Promote the procedure to the installed state: Promote

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

Chapter 17. Using CMVC

CMVC is an abbreviation for Configuration Management and Version Control.

Creating a Subsystem

Begin in the library that is to contain the subsystem.

1. Create a Command window: Create Command
2. Enter `cmvc.initial` and press Complete
3. At the `Subsystem` prompt, enter the name of the subsystem and press Promote

The command generates logging messages to the I/O window. When the command completes, the subsystem appears in the library. It contains an initial view called Rev1_Working.

Adding, Changing, or Deleting Ada Units in a View

Begin in the view's world (for example, Rev1_Working).

1. Go to the directory called Units.
2. Add, change, or delete Ada units as necessary.

Note: You cannot change controlled objects unless they are checked out.

Making Ada Units Controlled

Begin in the units directory for the view containing the units to be controlled.

1. Create a Command window: Create Command
2. Enter `cmvc.make_controlled` and press Complete
3. At the `What_Object` prompt, enter the wildcard ? and press Promote

The command generates messages to the I/O window. All units in the view are now controlled.

Note: If units are later added to the view, they will not be controlled unless you perform the above operations again.

Making a Subpath

Begin in the subsystem.

1. Place the cursor on the working view for the path from which the subpath is to be created (typically, Revn_Working).
2. Create a Command window: [Create Command]
3. Enter the command `cmvc.make_subpath` and press [Complete]
4. At the `New_Subpath_Extension` prompt, enter the name of the subpath (for example, the name of the developer who will be working in the subpath) and press [Promote]

The command displays messages in the I/O window. When it completes, a new view appears in the subsystem that is the working view for the subpath. This view has a name of the form *Pathname_Subpathname*_Working.

Checking Out a Unit for Changes

Begin in the unit to be changed.

1. Press: [Prompt For] - [Check Out]

   The check-out command appears in a Command window.
2. At the `Comments` prompt, enter the reason for the change and press [Promote]

The command displays its output in the Message window. When it completes, the unit can be modified.

Checking In a Unit after Changes

Begin in the unit to be checked in after changes.

1. Press: [Prompt For] - [Check In]

   The check-in command appears in a Command window.
2. At the `Comments` prompt, enter a summary of the changes made and press [Promote]

The command displays its output in the Message window. When it completes, the unit can no longer be changed and a new generation will have been created for the unit.

Making a Frozen Release

Begin in any library in the working view to be released or in the world for the working view. All controlled units in the view must be checked in.

1. Create a Command window: Create Command
2. Enter `cmvc.release` and press Promote

The command generates messages to the I/O window. When it completes, a new view, which is a frozen copy of the working view, appears in the subsystem world. The Environment automatically generates a release number. The form of the name of the released view is *Pathname/Subpathname_n_m*.

Note: Since the released view is frozen (units cannot be edited, promoted, and so on), be sure that the units in the working view are at the proper state (typically coded) before releasing.

Accepting Changes

Begin in the world of the view you want to make current.

If you do not want to accept any changes that will cause units in your view to be demoted

1. Place the cursor on the first line of the library display.

   Note: If you want to accept changes only for a specific unit, you can place the cursor on the library entry for the unit you want updated instead.
2. Accept changes: Accept Changes

The command displays its output to the Message window. All objects in the view are updated to the most current generation unless updating them causes demotions in your view.

If you want to accept all changes even if they cause units in your view to be demoted

1. Place the cursor on the first line of the library display.

   Note: If you want to accept changes only for a specific unit, you can place the cursor on the library entry for the unit you want updated instead.
2. Get the command to accept changes in a Command window: Prompt For - Accept Changes
3. At the `Allow_Demotion` prompt, enter true and press Promote

The command displays its output. All objects in the view are updated to the most current generation.

Getting Information

Determining out-of-date units in a view

Begin anywhere in the view.

1. Ask for a list of units that are out of date: Show Out Of Date

The list of units that are not the most recent generations available are displayed in the I/O window.

Determining units that are checked out in a view

Begin anywhere in the view.

1. Ask for a list of units that are checked out in the view: Checked Out In View

The units that are checked out in the view are displayed in the I/O window.

Determining units you have checked out (any view)

Begin anywhere in a view that defines the set of units that you may have checked out in that view or other views sharing its reservation tokens.

1. Ask for the units that you have checked out: Checked Out By User

A list of units you have checked out and the views to which they are checked out is displayed in the I/O window.

Change history for a unit

Begin in the unit of interest.

1. Create a Command window: Create Command
2. Enter `cmvc.show_history_by`, press Complete, and then press Promote

The history for the unit is displayed in the I/O window.

General information on a unit

Begin in the unit of interest.

1. Ask for information: Show Info

The command generates output to the I/O window. This output tells you what views share reservation tokens (this is, are subject to check-in/check-out synchronization of changes). It also tells you what generation of the unit you have and how many generations exist, who has the unit checked out, and so on.

Chapter 18. Networking

Logging Into Another System with Telnet

Begin in any window.

1. Create a Command window: [Create Command]
2. Enter `telnet.connect` and press [Complete]
3. At the `Remote_Machine` prompt, enter the name of the remote machine (enclosed in double quotes) and press [Promote]

Messages appear in the I/O window, the screen clears, and you are now connected to the remote machine and can begin logging in.

Interrupting a Telnet Session

Interrupting a Telnet session leaves the connection intact and takes you back to your original machine. You can later resume the interrupted session to continue work on the remote machine.

Begin in a Telnet session connected to a remote machine.

1. Interrupt the session: [Break]

The connection to the remote is interrupted and your original R1000 session reappears on the screen.

Note: If you are using a Rational Terminal and you are logged into another R1000 with Telnet, your terminal will be in Rational mode. In this mode the [Break] key is [Control][Meta][Mark].

Note: If the above steps do not work, the key that interrupts Telnet sessions may have been changed from [Break] to another key. Check with your system manager.

Resuming a Telnet Session

Begin in any window.

1. Create a Command window: [Create Command]
2. Enter `telnet.connect` and press [Complete]
3. At the `Remote_Machine` prompt, enter the name of the remote machine with which the connection was interrupted (enclosed in double quotes) and press [Promote]

The screen clears and the interrupted connection with the remote system is resumed.

You have to press the key that redraws the screen on the remote system (if the remote machine is another R1000, press [Control][L] to redraw the screen).

Terminating a Telnet Session

If you are still connected to the remote machine

1. Log off the remote machine.

For most remote Telnet servers, this terminates the Telnet session and returns you to your original session.

If you are not returned to your original session, interrupt from the session as described above and then follow the steps below.

If you are in your original R1000 session

Begin in any window.

1. Create a Command window: [Create Command]
2. Enter `telnet.disconnect` and press [Complete]
3. At the `Remote_Machine` prompt, enter the name of the remote system to which the session you want to terminate is connected.
4. Execute the command: [Promote]

The Telnet session is disconnected.

Copying a Single Object or Library onto Another R1000

Copying into an identical library structure keeping the same simple names for the items copied

Begin in the object or the library to be copied onto the other machine. Make sure that there are no selections in this window.

1. Create a Command window: Create Command
2. Enter `archive.copy` and press Complete
3. At the `Use_Prefix` prompt, enter the name of the machine onto which to copy prefixed with the string "!!"—for example, "!!m1".
4. Execute the command: Promote

The object and its children, or the library and its contents, are copied onto the designated machine in the same library structure and with the same names as on the source machine. Note that if the library structure does not already exist on the target machine, it is created automatically.

Copying into another library structure keeping the same simple names for the items copied

Begin in the object or the library to be copied onto the other machine. Make sure that there are no selections in this window.

1. Create a Command window: Create Command
2. Enter `archive.copy` and press Complete
3. At the `Use_Prefix` prompt, enter the name of the machine and the pathname of the target library to contain the object or library—for example, "!!m1!users-.sjl.example".
4. Execute the command: Promote

The object and its children, or the library and its contents, are copied onto the designated machine in the specified library structure and with the same names as on the source machine. Note that if the library structure does not already exist on the target machine, it is created automatically.

Copying Objects or Libraries from Another R1000

Copying into an identical library structure keeping the same simple names for the items copied

Begin in any window.

1. Create a Command window: Create Command
2. Enter `archive.copy` and press Complete
3. At the `Objects` prompt, enter the name of the machine and the pathname of the object from which to copy—for example, "!!m1!users.sjl.some_object".
4. Execute the command: Promote

The object and its children, or the library and its contents, are copied from the designated machine into the same library structure and with the same names as on the source machine. Note that if the library structure does not already exist on your machine, it is created automatically.

Copying into another library structure keeping the same simple names for the items copied

Begin in the library to contain the copied item.

1. Create a Command window: Create Command
2. Enter `archive.copy` and press Complete
3. At the `Objects` prompt, enter the name of the machine and the pathname of the target library to contain the object or library—for example, "!!m1!users.sjl-.example".
4. At the `Use_Prefix` prompt, enter `$`
5. At the `For_Prefix` prompt, enter the name of the library in which the object is located on the source machine without the machine name—for example, "!users.sjl".
6. Execute the command: Promote

The object and its children, or the library and its contents, are copied from the designated machine into the specified library structure and with the same names as on the source machine.

Copying Objects onto a Non-R1000 System

Begin in the object to be moved.

1. Create a Command window: [Create Command]
2. Enter `ftp.put` and press [Complete]
3. At the `To_Remote_File` prompt, enter the simple name (without a directory name prefix) of the object on the target system.
4. At the `Remote_Machine` prompt, enter the name of the remote machine (enclosed in double quotes).
5. At the `Username` prompt, enter your username on the remote machine (enclosed in double quotes).
6. At the `Password` prompt, enter your password on the remote machine (enclosed in double quotes).
7. If you want the object to go to a directory on the remote machine other than your home directory, at the `Remote_Directory` prompt, enter the full pathname of the directory to contain the object on the target (enclosed in double quotes).
8. Execute the Command: [Promote]

Copying Objects from a Non-R1000 System

Begin in the library to contain the object to be moved.

1. Create a command window: [Create Command]
2. Enter `ftp.get` and press [Complete]
3. At the `From_Remote_File` prompt, enter the simple name (without a directory name prefix) of the object on the remote system.
4. At the `To_Local_File` prompt, enter the name you want the object to have on your system.
5. At the `Remote_Machine` prompt, enter the name of the remote machine (enclosed in double quotes).
6. At the `Username` prompt, enter your username on the remote machine (enclosed in double quotes).
7. At the `Password` prompt, enter your password on the remote machine (enclosed in double quotes).
8. If the object on the remote machine is not in your home directory, at the `Remote_Directory` prompt, enter the full pathname of the directory on the remote machine containing the object.
9. Execute the command: [Promote]

RATIONAL