Rational R1000 Development System Series 200, 300, and 400

System Manager's Guide

RATIONAL

Product Number: 4000-00142

Rev. 3.0, November 1992

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.

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

Preface

INTENDED AUDIENCE

This guide provides instructions for management of the Rational R1000® Development System. It is useful for:

- System managers
- Operators
- Toolsmiths

ORGANIZATION

This guide is divided into sixteen chapters and four appendixes, as described in the following table:

Organization of the System Manager's Guide

Chapter/Appendix	Content
1: "Introduction to R1000 System Management"	Introduces system management, fundamental operations for getting started, and how to find the information you need.
2: "Stopping and Starting the System"	Describes ways to remove power from and restore power to the system in both normal and emergency situations. This chapter also describes how to halt the higher-level software of the Environment without removing power from the system.
3: "Managing User Accounts"	Describes commands to add and delete users, set up password policies, monitor user-account activity, customize user accounts, and transfer login sessions.
4: "Access Control"	Introduces access-control concepts and describes the commands used to control user access to files, Ada units, and worlds.
5: "Maintaining System Efficiency"	Describes the overall function of the five major system-daemon tasks and describes operations that can be used to maintain system efficiency.
6: "Tape Information and Operations"	Introduces tape formats and describes how to respond to tape-mount requests for reading and writing tapes.
7: "Saving and Restoring System Data"	Contains detailed, operation-specific descriptions of tape operations, including how to make a system backup, how to restore the Environment from backup, and how to archive and restore individual objects. This chapter also includes sections on safeguarding system information and sections describing less common tape operations.

Organization of the System Manager's Guide (continued)

Chapter/Appendix	Content
8: "System and Hardware Reports"	Describes how to generate system and hardware reports that are used to monitor system activity and spot possible trouble areas.
9: "Sending Messages"	Describes the mechanisms a system manager can use to communicate system information to users.
10: "Terminal Information"	Describes terminal types and terminal-type objects. Gives instructions for creating custom key bindings.
11: "Series 200 System Overview"	Introduces the hardware components of the Series 200 system.
12: "Series 300 System Overview"	Introduces the hardware components of the Series 300 system.
13: "Series 400 System Overview"	Introduces the hardware components of the Series 400 system.
14: "Printer Operations and Maintenance"	Describes the Rational line printer, including how to operate and maintain it.
15: "Tape Drives"	Describes the 9-track and 8-millimeter tape drives, including maintenance and cleaning operations.
16: "Disk Drives"	Describes disk-drive operations and diagnostics.
A: "Commands for the Operator's Console Command Interpreter"	Summarizes the commands a system manager is most likely to execute from the operator console using the language of the operator's console command interpreter.
B: "Diagnostic Crash Procedures for Release D_12_7_3 (Delta 3.1)"	Provides procedures that can be used to save the system state, make a crash-dump tape, run diagnostic tests to pinpoint the cause of the crash, and attempt to reboot the machine.
C: "Diagnostic Crash Procedures for Release D_12_6_5 (Delta 3.0)"	Provides procedures that can be used to save the system state, make a crash-dump tape, run diagnostic tests to pinpoint the cause of the crash, and attempt to reboot the machine.
D: "Rational Customer-Support Services"	Describes the Rational support model and how to communicate with support organizations. At many sites, the system manager can be the single point of contact with Rational support organizations. It may be the responsibility of the system manager to decide which system or software problems should be forwarded to Rational technical support personnel.
E: "Machine Initialization"	Describes the initialization software that runs at boot time and gives instructions on modifying this software.
F: "Quick Reference for Parameter-Value Conventions"	Summarizes the Environment-defined conventions for parameter values in commands.

ASSOCIATED DOCUMENTS

For information about using the *Rational Environment Reference Manual*, see the Introduction to the Documentation Set in the Reference Summary (RS) book. The Reference Summary also contains a glossary and a master index.

For an introduction to the Environment, see the *Rational Environment User's Guide.*

For detailed information on the Ada language, see the *Reference Manual for the Ada Programming Language.*

Many of the tasks described in the *System Manager's Guide* (SMG) use commands that are documented in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, as shown in the following table:

SMG and SMU

Task Described in SMG	**SMU Packages**
Stopping and starting the system	Operator
Maintaining system efficiency	Daemon
Managing user accounts and sessions	Operator
Controlling membership to access groups for files, Ada units, and worlds	Operator
Sending messages to users	Message
Executing system backups, tape operations, and restoration using tape	System_Backup, System_Utilities, Tape
Entering common commands from the operator console	Message, Operator, Queue, Terminal

Finally, software contained in !Machine.Initialization is related to packages Daemon, Operator, Queue, Scheduler, and Terminal; this software initializes system parameters during reboot and in many cases provides a preferred alternative to certain parameter-setting procedures otherwise handled by the above packages. See the online documentation in !Machine.Initialization for more information.

Contents

1 Introduction to R1000 System Management

This chapter introduces the following aspects of R1000® system management:

- An overview of the system manager's duties
- Access to the Rational Environment™, including the devices the system manager can use to enter commands
- The operator's console interfaces, including the different user interfaces available for entering commands from the operator's console
- Conventions used in this manual for different kinds of command prompts

Each section tells you where to find more detailed information.

SYSTEM MANAGER'S DUTIES

The specific duties of the system manager include:

- Managing system access
- Managing system resources
- Maintaining system availability
- Coordinating operator activities
- Communicating with the users
- Communicating with Rational technical representatives

This section provides a summary of these duties. Refer to *Rational Environment Training: System Management* for more information, including comprehensive system-management training.

Managing System Access

System managers are responsible for creating new user accounts and removing account access. They also establish policies for controlling access to system information, including defining *access-control groups* and allocating *operator capability.*

Managing System Resources

System managers are responsible for:

- Checking the status of the system
- Scheduling the system daemons

- Managing disk space and the job scheduler
- Configuring the Environment for networking, mail, and product registration

Maintaining System Availability

System managers are responsible for:

- Configuring terminal ports for login, printers, and Cross-Development Facility (CDF) interfaces
- Maintaining the network configuration
- Ensuring access to external modems
- Resolving hardware failures

System managers also may be responsible for the basic hardware configuration and maintenance of printers, terminals, workstations, tape drives, and disk drives connected to the system.

Coordinating Operator Activities

Many system managers are responsible for coordinating the activities of operations personnel, who perform tasks such as:

- Handling tape-mounting requests
- Preserving system data through archives and backups
- Stopping and restarting the system
- Handling emergency situations

Communicating with Users

System managers are responsible for handling Environment problem reports from users and informing users about hardware failures. Problems that can be handled directly by the system manager include responding to known bugs with workarounds, handling misconceptions about how the Environment should work, and deciding how to best use the facilities to approach a problem.

It is important that system managers provide feedback to users about their reported problems. Without feedback, users become reluctant to continue submitting problems.

System managers are also responsible for notifying users of:

- Scheduled backups
- Scheduled system downtime
- Availability of new tools and commands

Communicating with Rational Technical Representatives

System managers are usually responsible for communicating with Rational personnel about problems and failures that cannot be handled directly by the system manager. Some problems are more complicated than they seem at first. The Response Center also may be able to suggest a workaround if there actually is a bug.

ACCESS TO THE RATIONAL ENVIRONMENT

This section describes the ways in which a system manager can gain access to the Rational Environment through different terminal configurations. For details related to terminal configuration, such as how to set up terminal display features, character sets, and keyboard features, see the user's guide for your terminal, terminal emulator, or workstation.

See Appendix E and the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, package Terminal, for complete information and procedures for changing port settings. See Chapter 10 for complete information about changing terminal types and key bindings.

Classes of Terminals Connected to the R1000

Three classes of terminals provide access to the Rational Environment, as shown in Figure 1-1.

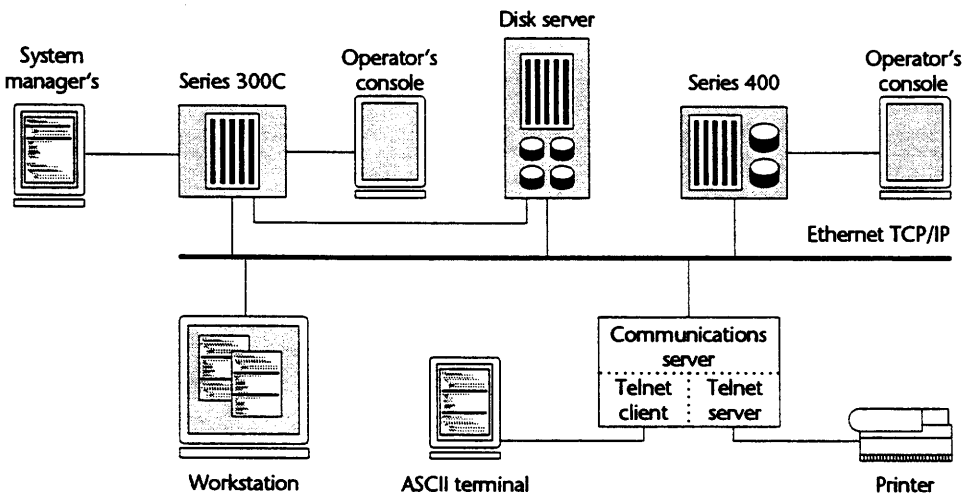

Figure 1-1 Access to the Environment

- *Operator's console:* Provides a hard-wired, line-oriented, low-level interface to the Environment. This terminal must be VT100™-compatible, and it is connected to an RS232 port labeled "operator console." Certain commands can be entered *only* from the operator's console.
- *System manager's Environment terminal:* Provides an optional, hard-wired, screen-oriented interface for Environment sessions. This terminal is directly connected to a specific RS232 port (described below), or it can be omitted so that other devices can be connected to this port.
- *Users' Environment terminals:* Provide standard user access to Environment sessions. Most system-management tasks can be performed from these terminals, replacing the need for the system manager's Environment terminal in most cases. Users' Environment terminals are not available for Environment access, however, unless the connecting R1000 ports are enabled for login and the communications circuit is functioning (for example, as provided by a network circuit).

Each terminal class is described in more detail below.

Operator's Console

The required operator's console is a VT100-compatible terminal or terminal emulator that is connected to an RS232 communications port labeled "operator console." (See Figures 11-2, 11-5, 11-7, 12-2, 12-8, and 13-2 for the location of this port.)

The operator's console must be provided by the customer at the time of an R1000 installation. This console must be at least a 9600-bits/sec, ANSI-standard, VT100-compatible terminal.

The operator's console is usually installed in the computer room near the R1000; this allows an operator to mount tapes and execute commands without leaving the computer room.

From the operator's console, you can enter commands to:

- Halt the system using the [Break] key
- Monitor system activity
- Shut down the system for scheduled downtime
- Initiate the boot process
- Schedule automatic system "housekeeping" operations
- Create and delete user accounts and access-control groups
- Perform system backups
- Restore the system from backup tapes
- Archive files and Ada units
- Create print queues
- Configure ports
- Handle tape-mount requests
- Run diagnostics

These operator's console functions are executed through:

- The operator's console command interface from the `Command:` prompt
- The Environment Elaborator Database (EEDB) from the `EEDB:` prompt
- The Environment kernel from several different prompts:
  - The low-level Environment `Kernel:` prompt
  - The preferred `Kernel:` prompt available from the EEDB
  - The privileged `*Kernel:` prompt available from the EEDB
- The Command Language Interpreter (CLI) from the `CLI>` prompt

These interfaces are described in more detail in the "Operator's Console Interfaces" section, below.

System Manager's Environment Terminal

A specific RS232 communications port on the R1000 can be used for directly connecting the system manager's Environment terminal. This port is commonly known as *port 16*, but it is labeled differently for each R1000 series:

- *On the Series 400,* the port labeled "comm port" can be configured as port 16. This port is located on the I/O panel, behind the front door of the cabinet (see Figure 13-2).

- *On the Series 300S and Series 300C,* the port labeled "external modem" can be configured as port 16. This port is located on a chassis panel, behind the front door of the cabinet (see Figures 12-2 and 12-8).
- *On the Series 200,* port 16 is the leftmost port located on the communications panel mounted near the bottom of the chassis, behind the front panel (see Figures 11-2, 11-5, and 11-7).

Port 16 can be used to access the Environment directly from an Environment terminal. This terminal must be at least 9600-bits/sec, ANSI-standard, VT100-compatible. Your site will need to acquire and install the proper cables for connecting the Environment terminal to the R1000. You may have to customize your own keymap to properly access the Rational Environment from this terminal (see Chapter 10). The Environment terminal is usually installed outside the computer room to allow remote access for the system manager.

If this port is not being used for direct access to the Environment by a customer-provided Environment terminal, it can be connected to a customer-provided modem. Using this modem, Rational support personnel can access the customer's Environment and perform remote diagnostics (this applies only to customers who have purchased standard support). See Appendix D for more details about Rational's customer-support services.

If this port is not connected to an Environment terminal or to an external modem, it can be used to connect to a printer.

Users' Environment Terminals

Most users access the Environment from a terminal through a communications server or from a terminal emulator on either a workstation using the X Window System™ (through Rational's RXI product) or an IBM-compatible PC using Microsoft Windows™ (through Rational's RWI product).

Rational provides terminal emulators and key bindings for the following:

- Apollo DN3000
- ASCII terminals
- DEC ULTRIX™ workstation
- DEC VAXstation™ VMS™
- HP 9000
- IBM PC-compatible with 86-, 91-, or 101-key keyboard
- IBM RISC System/6000™
- IBM Xstation 120
- NCD™ terminal
- Sun Workstation®

THE OPERATOR'S CONSOLE INTERFACES

This section describes the different user interfaces with which the system manager interacts with on the operator's console and the different prompts displayed within these interfaces. The three major user interfaces available from the console are:

- The low-level R1000 kernel
- The Environment Elaborator Database (EEDB)
- The operator's console command interpreter

In addition to these three major interfaces, the console provides:

- The preferred kernel, available through the EEDB
- The privileged kernel, available through the EEDB
- The Command Language Interpreter (CLI), available only when the system has been stopped or has crashed

Cycling through the Three Major Interfaces

To get to each major console interface, press [Control] [Z] repeatedly. As you cycle to each interface, a title of the form:

```
====>> Some Title <<====
```

appears above the current prompt. On a fully elaborated, properly functioning system, it takes three presses of [Control] [Z] to return to the interface from which you started the cycling process. The titles for the three interfaces are:

```
====>> Kernel.x.y.z <<====

====>> Elaborator Database <<====

====>> Console Command Interpreter (System Job n) <<====
```

where *x*, *y*, and *z* are numbers representing the current version of the kernel and *n* is the number of the job associated with the interpreter.

Three Different Kernel Prompts

The kernel is a low-level interface to the operating system. Through the kernel you can gather information concerning job states, volume state, medium-term scheduler states, and error logging. (The kernel also has a privileged command mode, in which the `*Kernel:` prompt is displayed. The commands in this mode are intended for Rational's use and may cause irreparable damage to the Environment when used improperly.)

There are three ways to access the kernel:

- From the preferred EEDB `Kernel:` prompt, accessible by executing the command `kernel` from the `EEDB:` prompt as follows:

  ```
  ====>> Elaborator Database <<====
  EEDB: kernel
  Kernel:
  ```

  Note: *This is the preferred way to access the kernel when executing most kernel commands.*

- From the privileged EEDB `*Kernel:` prompt, accessible as follows:

  1. Execute the `kernel` command at the `EEDB:` prompt:

     ```
     ====>> Elaborator Database <<====
     EEDB: kernel
     ```

  2. Execute the `enable_priv_cmds` command at the `Kernel:` prompt:

     ```
     Kernel: enable_priv_cmds
     ```

     A warning message is displayed, and then you are prompted for instructions to proceed or not.

3. If you choose to proceed, enter `true` at the prompt, and then enter a password when asked. The `*Kernel:` prompt appears:

```
Proceed [FALSE]: true
Password: <password>
*Kernel:
```

Caution: *The *Kernel contains a set of commands that must be used with extreme care by knowledgeable support personnel only. These commands can easily crash or hang the machine; some can corrupt the state of the machine so that you must recover the machine from backup tapes. Use the *Kernel only when explicitly instructed to do so by this guide or a Rational support representative.*

- From the low-level R1000 `Kernel:` prompt appearing on the console as:

```
====>> Kernel.11.5.7 <<====
Kernel:
```

Caution: *You should usually enter all kernel commands from the EEDB's kernel rather than the low-level R1000 kernel. Certain commands can cause a page-fault error condition when executed from the low-level R1000 kernel after the Environment has been elaborated. Use the low-level R1000 kernel only when explicitly instructed to do so by this guide or a Rational support representative.*

Note: *If you see the `Kernel:` prompt on the console but are uncertain whether it belongs to the EEDB kernel or the R1000 kernel, cycle through all of the available operator's console interfaces as described in the preceding subsection "Cycling through the Three Major Interfaces." When you get to the EEDB `kernel:` prompt, the "Elaborator Database" banner appears.*

Note: *To return to the EEDB from either the preferred or the privileged EEDB `Kernel:` prompts, enter the `quit` command.* ***Do not****, however, enter the `quit` command from the low-level R1000 kernel.*

The Environment Elaborator Database (EEDB) Prompt

The Environment Elaborator Database maintains information on Environment configurations and their subsystems. This interface is used by the R1000 during the boot process to elaborate the Environment.

Two forms of the kernel interface are also available from the EEDB: preferred and privileged. The EEDB kernel interface should be used instead of the R1000 kernel interface after the Environment has been elaborated—otherwise, certain commands can page fault, crashing the system.

The Operator's Console Command Interpreter

Most system-management functions can be executed from the operator's console using the *operator's console command interpreter.* Note that this interface is not intended for software development. See Appendix A for a summary of the most important commands you can execute from this command interpreter.

Installing the Command Interpreter

The operator's console command interpreter is distributed as an Environment program, located in the !Tools.Ci subsystem. Always use the most recent version

(view) of this subsystem. The !Machine.Release.Current.Activity must include an entry for !Tools.Ci, specifying the most recent spec and load view. This activity is set correctly when shipped to you. If necessary, it can be reset with the Activity-.Add command.

The operator's console command interpreter is started by machine initialization (see Appendix E) when the system boots. If machine initialization fails to start the operator's console command interpreter, you must execute the Ci.Login procedure through the following command:

```
Program.Run_Job (Program.Current("!Tools.Ci", "Ci.Login",
        Activity => "!Machine.Release.Current.Activity"),
        Options  => "Name => (Console Command Interpreter)",
        Context  => "!Machine.Error_Logs");
```

Input to the Command Interpreter

Each line of input to the operator's console command interpreter is treated as an Ada program. If multiple Environment commands are entered on a single line and are separated by semicolons, they will be executed in the sequence entered. In general, any valid Ada statement (conditionals, loops, and the like) can be used, limited only by the line length (256 characters). The Def, Quit, and Help commands are exceptions: each should be entered on a separate line.

[Control][G] causes the current foreground job to run as a background job. Additional commands then can be executed. Output from each running job is identified with a job ID header.

Restrictions and Limitations

- Commands that create displays edited with commands from package Common (for example, What.Jobs and Links.Edit) do not work when they are run from the operator's console command interpreter. If you try to execute them, an error message is displayed and the command does not execute.
- Commands that use window I/O do not work when they are run from the operator's console command interpreter.
- The maximum command-line length is 256 characters.

Logging Into the Command Interpreter

To log into the operator's console command interpreter:

1. From the operator's console, press [Control][Z] until the following prompt appears:

   ```
   ====>> Console Command Interpreter (System Job 244) <<====
   username:
   ```

   Note: *See "Cycling through the Three Major Interfaces," above, for more information on using* [Control][Z].

2. Enter a username, password, and session:

   ```
   ====>> Console Command Interpreter (System Job 244) <<====
   username: operator
   password:
   session: S_2
   ```

   Note: *The password is not echoed.*

If you successfully log in, the console displays a log message and then the command interpreter's `command:` prompt appears:

```
92/08/12 19:13:58 --- operator.s_2 logging in.

====>> Ci.Interpret (OPERATOR.S_2 Job 212) <<====
command:
```

Once you have the `command:` prompt, you are able to enter Environment commands as described below in "Conventions Used in This Manual for Command Prompts." The context is initially set to the home world of the username supplied during login, and command names are resolved using the user's session searchlist. For a description of commands commonly entered from the command interpreter, see Appendix A.

Note: *If you do not enter input within the timeout period specified by !Machine-.Initialization.Rational.Servers (usually 60 seconds), you are automatically logged off the command interpreter.*

Editing Functions Available from the Command Interpreter

You can edit input to the command interpreter using the key shown in Table 1-1.

Table 1-1 Command-Interpreter Editing Functions

Function	Key
Delete the previous character (backspace)	[Delete] or [Control][H]
Delete the current line of input	[Control][U]
Replace the current line of input with the previous line of input	[Control][A] or [Control][B]

Logging Off the Command Interpreter

To log off the operator's console command interpreter, execute the command:

```
====> Ci.Interpret; (Operator.S_2 Job 212) <====
command: quit
```

The username prompt reappears:

```
====>> Console Command Interpreter (System Job 244) <<====
92/08/12 19:56:19 --- operator.S_2 logged out.
username:
```

Note: *If you do not enter input within the timeout period specified by !Machine-.Initialization.Rational.Servers (usually 60 seconds), you are automatically logged off the command interpreter.*

The Command Language Interpreter (CLI) Prompt

The command language interpreter (CLI) interface is implemented by the input/output controller (IOC) and provides the lowest level of system interaction.

Note: *You have access to the `CLI>` prompt only after the system has been stopped or has crashed.*

You use the CLI primarily to run hardware diagnostics and start the boot process after the system has crashed. When you attempt to restart the system, the following menu appears:

```
Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

This menu provides access to the CLI. Instructions on using the CLI are given within the context of the procedures that require the CLI.

Note: *Do not confuse the CLI with the operator's console command interpreter, a line-oriented interface for entering Environment commands at the console.*

CONVENTIONS USED IN THIS MANUAL FOR COMMAND PROMPTS

System-management operations are executed from either:

- An Environment session command window on either the system manager's Environment terminal or a user's Environment terminal
- Prompts appearing on the operator's console

If you need more information about using Environment command windows, refer to both the *Rational Environment User's Guide* and *Rational Environment Basic Operations.*

Note: *Although many of the system-management commands can be executed from either an Environment command window or the operator's console, some tape-related commands can be executed only from the operator's console command interpreter.*

Commands without a Prompt

If an instruction step in a task does not make a distinction about where a command should be entered, the command can be entered from either the operator's console command interpreter or an Environment command window.

Examples that show how the command and its parameters should be entered are depicted in the simplest format: without console command prompts, Environment command-window completions, or trailing semicolons. For example, suppose an instruction step asks you to execute the Sample_Environment.Action command with the following parameters:

```
sample_environment.action(123,"test")
```

Examples that emphasize parametric interfaces show commands as they appear after completion in an Environment window, including the terminating semicolon. For example, the following illustrates the parametric interface of the Operator-.Create_User command:

```
Operator.Create_User (User     => ">>USER NAME<<",
                      Password => "",
                      Volume   => 0,
                      Response => "<PROFILE>");
```

Commands with a Prompt

When an instruction step in a task explicitly states that a command must be entered from the operator console, examples showing how the command should be entered by the user also depict the appropriate prompt.

Example of the `command:` Prompt

If an instruction step asks you to execute a command from the console's `command:` prompt, an example showing how the command is entered appears as:

```
command: sample_occi_command("test")
```

Example of the `EEDB:` Prompt

If an instruction step asks you to execute a command from the console's `EEDB:` prompt, an example showing how the command is entered appears as:

```
EEDB: EEDB_Restricted_Action
```

Examples of Three Different Kernel Prompts

There are three different kinds of kernel prompts:

- The preferred `Kernel:` prompt available from the EEDB

  If an instruction step asks you to execute a command from the EEDB preferred `Kernel:` prompt, an example showing how the command is entered appears as:

  ```
  ====>> Elaborator Database <<====
  Kernel: EEDB_Kernel_Restricted_Action
  ```

- The privileged `*Kernel:` prompt available from the EEDB

  If an instruction step asks you to execute a command from the EEDB privileged `*Kernel:` prompt, an example showing how the command is entered appears as:

  ```
  ====>> Elaborator Database <<====
  *Kernel: Dangerous_Kernel_Action
  ```

  Caution: *The *Kernel contains a set of commands that must be used with extreme care by knowledgeable support personnel only. These commands can easily crash or hang the machine; some can corrupt the state of the machine so that you must recover the machine from backup tapes. Use the *Kernel only when explicitly instructed to do so by this guide or a Rational support representative.*

- The low-level Environment `Kernel:` prompt

  If an instruction step asks you to execute a command from the console's low-level Environment `Kernel:` prompt, an example showing how the command is entered appears as:

  ```
  ====>> Kernel.11.5.7 <<====
  Kernel: Kernel_Restricted_Action
  ```

  Caution: *You should usually enter all kernel commands from the EEDB kernel rather than the low-level R1000 kernel. Certain commands can cause a page-fault error condition when executed from the low-level R1000 kernel after the Environment has been elaborated. Use the low-level R1000 kernel only when explicitly instructed to do so by this guide or a Rational support representative.*

2 Stopping and Starting the System

This chapter describes the procedures for normal and emergency system shutdown and reboot. If your system has crashed, see Appendix C for procedures that diagnose the cause of the crash and provide for recovery.

OVERVIEW

There are two parts to stopping the Rational system:

- Shutting down the Environment software, which can be done in two ways:
  - Normal shutdown using the !Commands.Abbreviations.Schedule_Shutdown command
  - Deliberate crash using the [Break] key
- Powering off the R1000, which can be done in two ways:
  - Normal power-off
  - Emergency power-off

In parallel, there are two parts to starting the system:

- Powering up the R1000
- Booting the Environment

This chapter discusses:

- *Normal Environment shutdown,* which is used for all scheduled shutdowns, including preparation for powering off the R1000
- *Normal power-off,* which removes power from the system after a normal Environment shutdown
- *Deliberate crashing of the Environment,* which is used when the normal shutdown procedure cannot be followed
- *Powering up the system,* which is used after normal or emergency power-off
- *Standard boot process,* a description of the four phases of the boot process and their messages
- *Emergency power-off,* which removes power from the system as quickly as possible

Note: *The "Emergency Power-Off" section is at the end of this chapter, behind a red tab.*

NORMAL ENVIRONMENT SHUTDOWN

To stop the Environment without risking loss of data, use the !Commands.Abbreviations.Schedule_Shutdown command. This is the recommended method for shutting down the Environment because the Shutdown command automatically:

- Issues several warnings to users. The first warning occurs when the command is executed, the next occurs after 3/4 of the interval has passed, the next occurs when 3/4 of the remaining time has passed, and so on, until the Environment is shut down. Note that a warning interval of 30 seconds or less results in immediate shutdown.
- Logs off all users who have not already done so and disables terminal ports.
- Aborts all running jobs.
- Takes a snapshot of the current Environment state so that the system can be restored to that state when rebooted.
- Updates the system error log with the date, time, reason for the shutdown, and any comments.

Executing the Schedule_Shutdown Command

Note: *The following steps require operator capability. (See "Special Groups" in Chapter 4.)*

To shut down the Environment in a nonemergency situation:

1. Warn all users of the planned shutdown. Give this warning before you execute step 4, and be sure to allow users enough time to save uncommitted objects. (Step 4 sends additional warnings automatically, but users who log in later may miss one or more of these messages. Use mail or another method to inform users of a planned shutdown.)
2. Locate the operator-mode keyswitch on the control panel (see Figure 2-1, 2-2, or 2-3).
   - If you want to perform an *automatic reboot* and do not want to power down the system, turn this keyswitch to the Automatic position.
   - If you want to perform an *interactive reboot* or if you want to power down the system, turn this keyswitch to the Interactive position.
3. Determine the reason code and an explanation for the planned shutdown. Both the reason code and an explanation are entered when you execute the Schedule_Shutdown command in step 4.

   If you want to see the currently available reason codes, execute the following command:

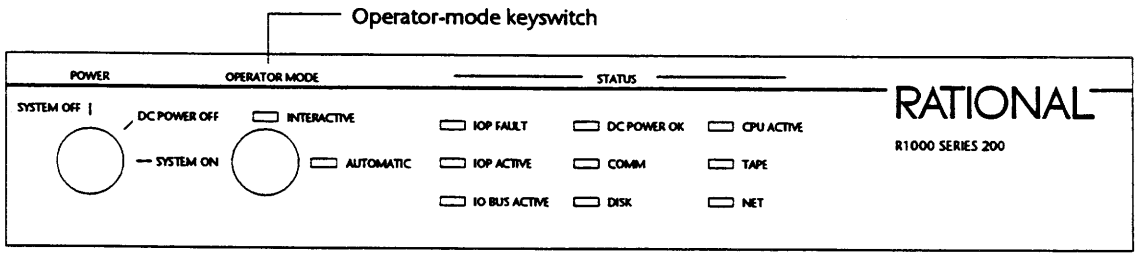

Figure 2-1 Series 200 Control Panel: Operator-Mode Keyswitch

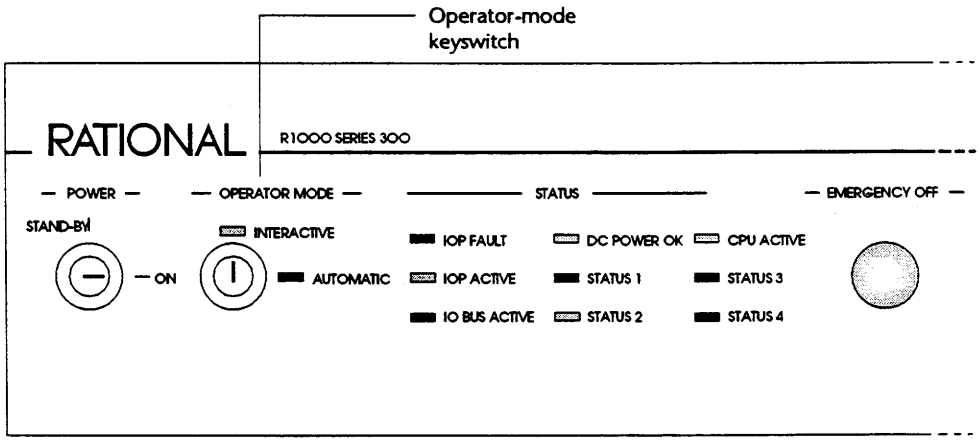

Figure 2-2 Series 300 Control Panel: Operator-Mode Keyswitch

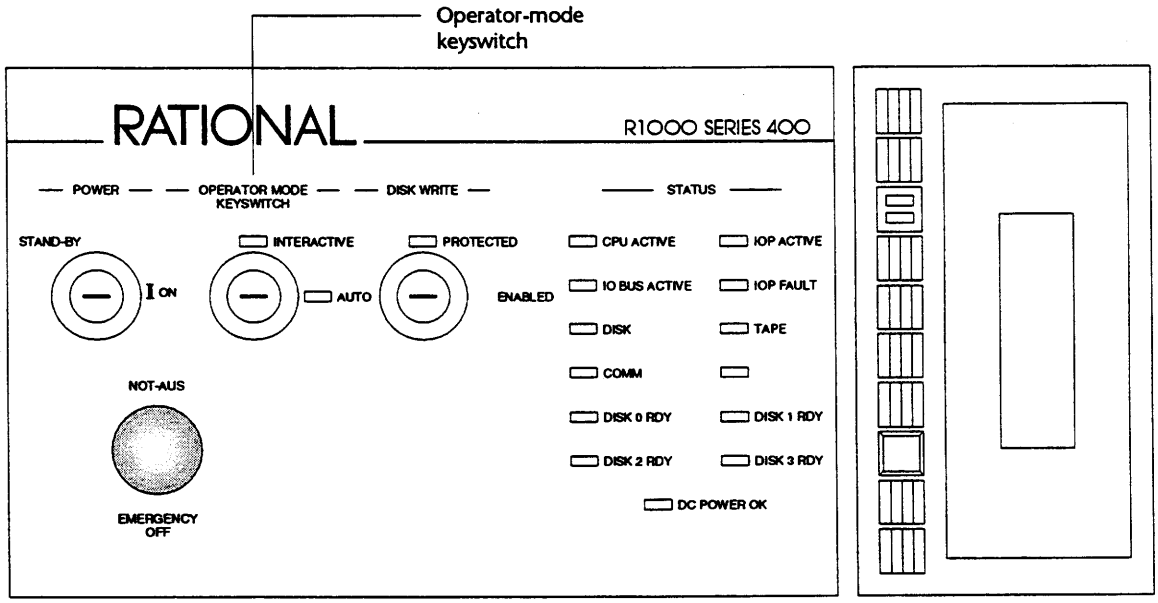

Figure 2-3 Series 400 Control Panel: Operator-Mode Keyswitch

```
Operator.Shutdown("?")
```

This displays the following:

```
Enter one of the following shutdown codes.
Then enter any comments.

Cops    - Customer Operations
Release - Loading of new release
Maint   - Scheduled Maintenance
Crash   - System Crashed
Hang    - System was hung
Other   - Other reason.  Give details in comments
```

Your explanation can include other relevant information, such as the Environment state at the time of shutdown.

4. Execute the !Commands.Abbreviations.Schedule_Shutdown command, specifying:
   - Shutdown time in 24-hour date format with an optional date (YR/MO/DA HR:MIN:SEC)

- Shutdown code as explained in the previous step, for example: `Maint`
- Comments, for example: `AC Work`

For example, assuming that the shutdown is to take place at 3:30 P.M., enter the command:

```
Schedule_Shutdown ("15:30", "Maint", "AC Work")
```

Note: *Scheduling a shutdown time that is 30.0 seconds or less from the time the command is executed results in an immediate shutdown.*

To cancel a shutdown at any time *before* the scheduled shutdown time, enter the command:

```
Operator.Cancel_Shutdown
```

Note: *You cannot reliably cancel a shutdown after the shutdown starts.*

5. After the shutdown completes, the next action depends on the setting of the operator-mode keyswitch.
   - If the operator-mode keyswitch is in the Automatic position, the system reboots as described in "Description of the Standard Boot Process."
   - If the operator-mode keyswitch is in the Interactive position, the boot process pauses at the following menu:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

   - If you intend to power-off the system, *do not type anything at the menu's prompt*; instead, continue as described in the next section, "Normal System Power-Off."
   - If you intend to reboot the system without powering off, press [Return]. the system reboots as described in "Description of the Standard Boot Process."

NORMAL SYSTEM POWER-OFF

Perform a normal system power-off before:

- Performing a scheduled maintenance
- Moving the system to another location
- Performing a hardware upgrade

To power-off the system in a nonemergency situation:

1. Shut down the Environment as described in the preceding section, "Normal Environment Shutdown." At the end of that procedure, the following menu prompt should appear:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

Note: *Do not enter anything—leave the boot process in this state since you want to power-down the system.*

2. Locate and activate the write-protect (PTCT) switch(es) for the disk(s):

   Note: *Some older models of disks may not have a write-protect switch or the position of the switch may vary.*

   - On the Series 200:

     If your system is a Series 200 Model 10, Model 20B, or Model 40B, see Figure 2-4. If your system is a Series 200 Model 20 or a Model 40, see Figure 2-5.

     Switch *each* drive's write-protect switch to the ON position to enable write protection.

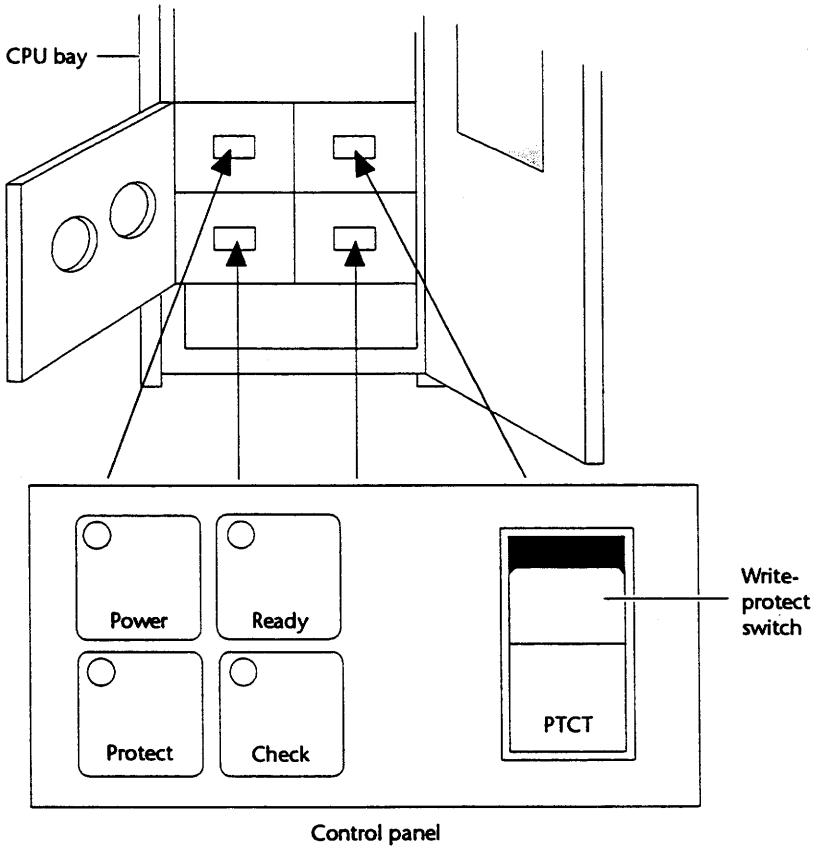

Figure 2-4 Disk Write-Protect Switches for Series 200, Models 10, 20B, and 40B

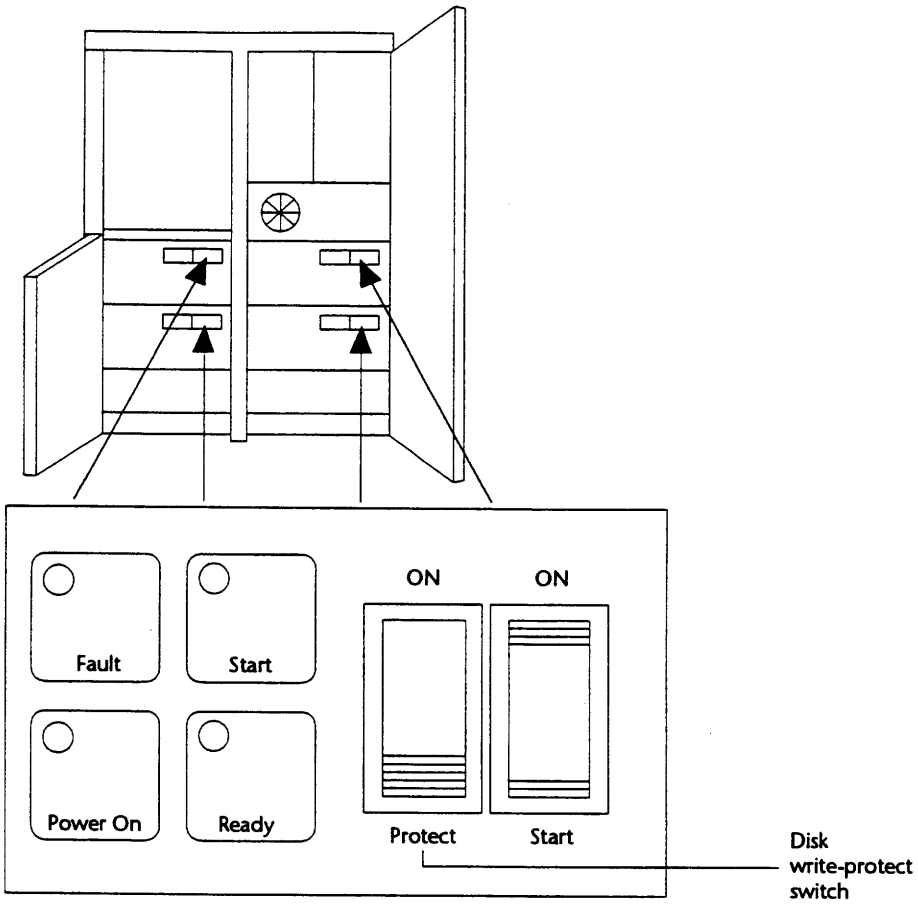

Figure 2-5 Disk Write-Protect Switches for Series 200, Models 20 and 40

- On the Series 300S only:

  Switch *each* drive's write-protect switch to the ON position to enable write protection (see Figure 2-6).

- On the Series 400:

  Write-protect the disks by turning the keyswitch labeled Disk Write to the Protected position (see Figure 2-7).

3. Turn the power keyswitch off. This switch is located on the system control panel (see Figure 2-8, 2-9, or 2-10) and the off position is marked System Off on the Series 200, and Standby on the Series 300 and 400.

   This step disconnects all AC and DC power from the cabinet (except AC power to the PDU and cooling fans on the Series 200).

4. To remove all power from the cabinet:

   a. For the Series 200 only: let the fans run several minutes to cool the components inside the cabinet.

   b. Switch the main circuit breaker CB1 (see Figure 2-11, 2-12, or 2-13) to the OFF position.

   c. Unplug the cabinet's AC power cord from the AC power source.

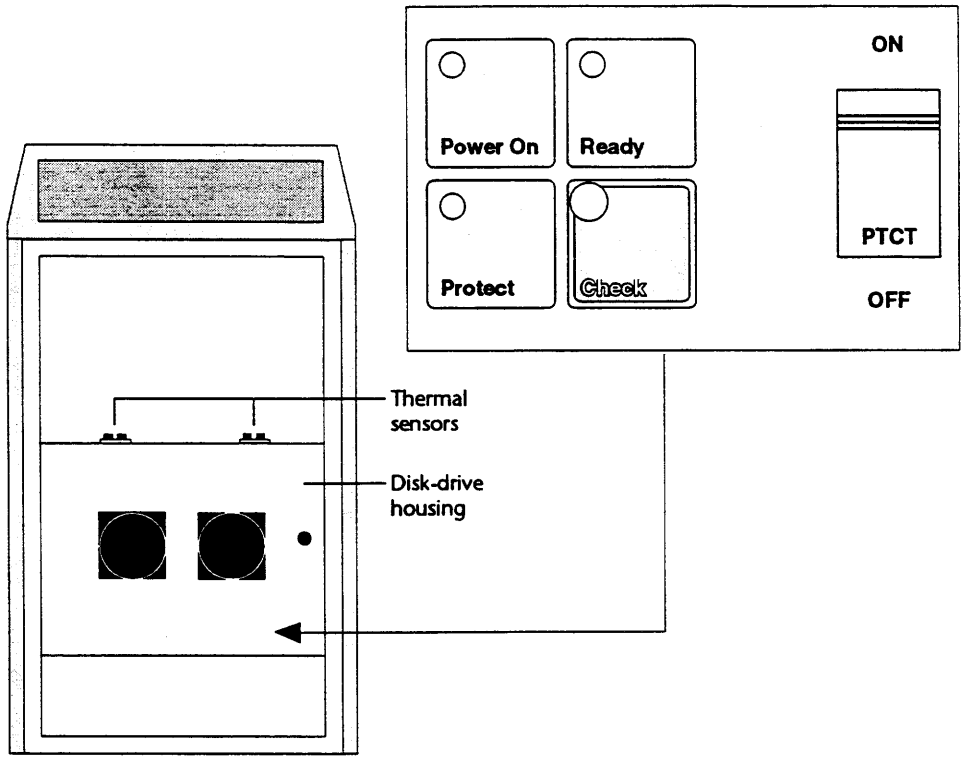

Figure 2-6 Series 300S, Peripheral Front View: Location of Disk-Drive Housing and Write-Protect Switch

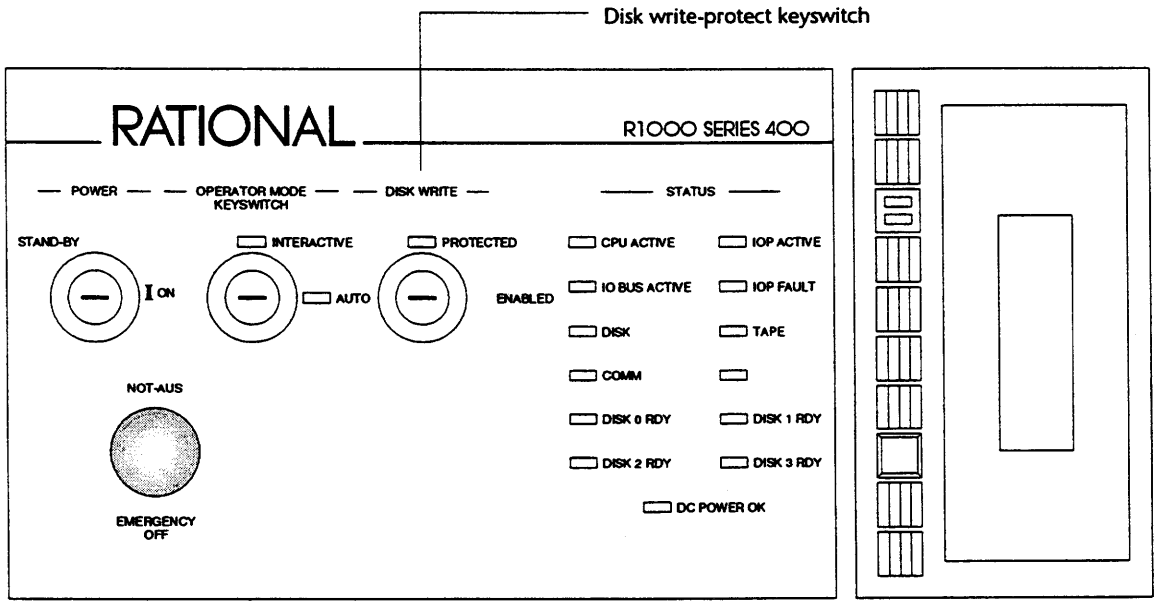

Figure 2-7 Control Panel for Series 400, Showing Write-Protect Switch

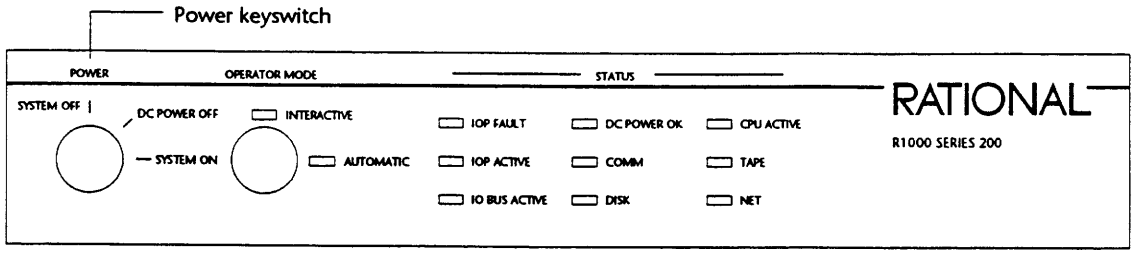

Figure 2-8 Series 200 Control Panel: Location of Power Keyswitch

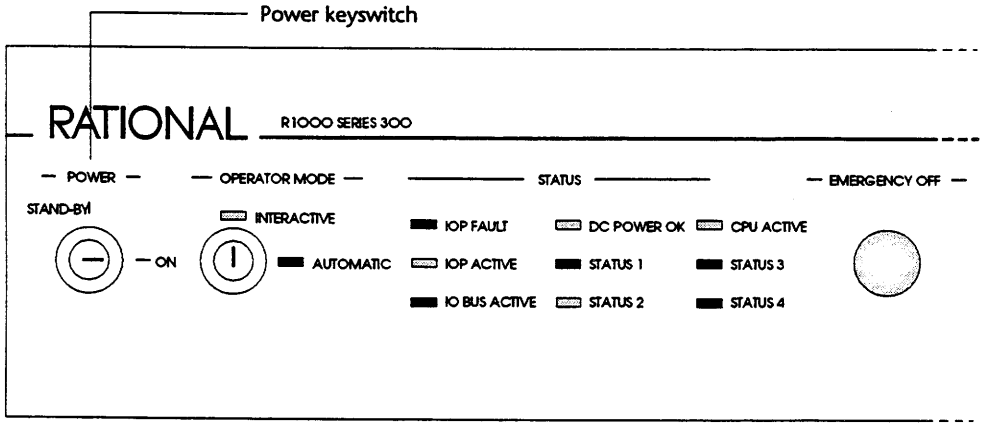

Figure 2-9 Series 300 Control Panel: Location of Power Keyswitch

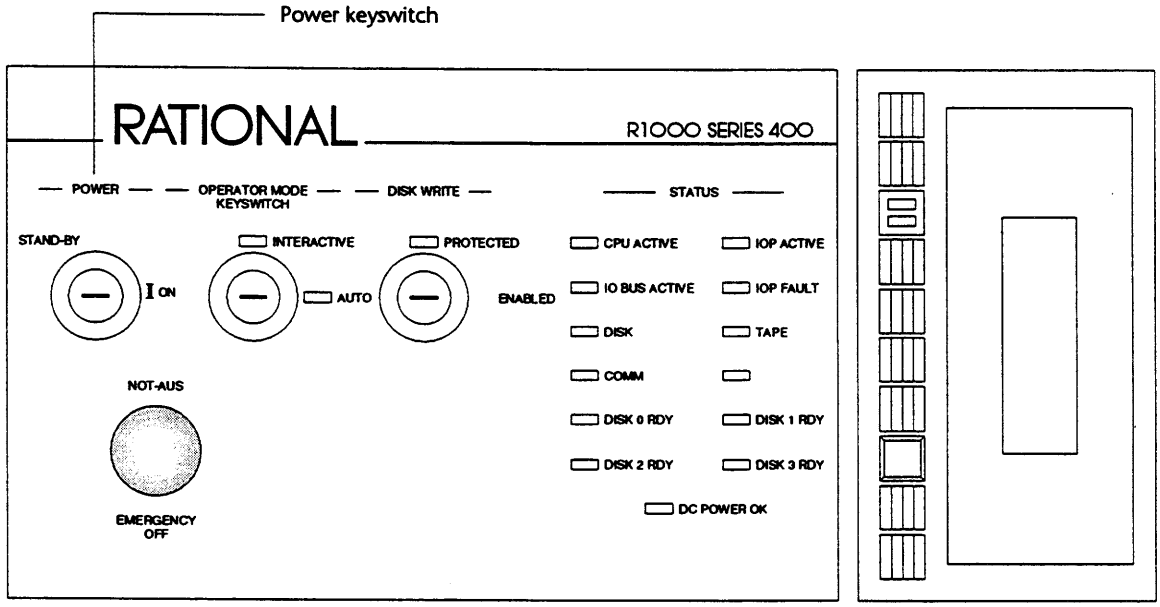

Figure 2-10 Series 400 Control Panel: Location of Power Keyswitch

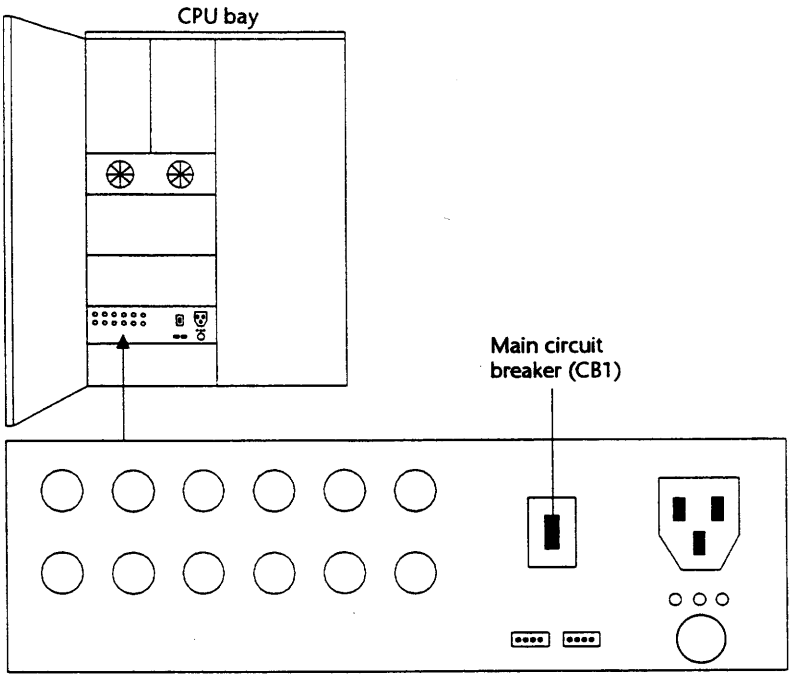

Figure 2-11 Series 200: Location of CB1 Circuit Breaker

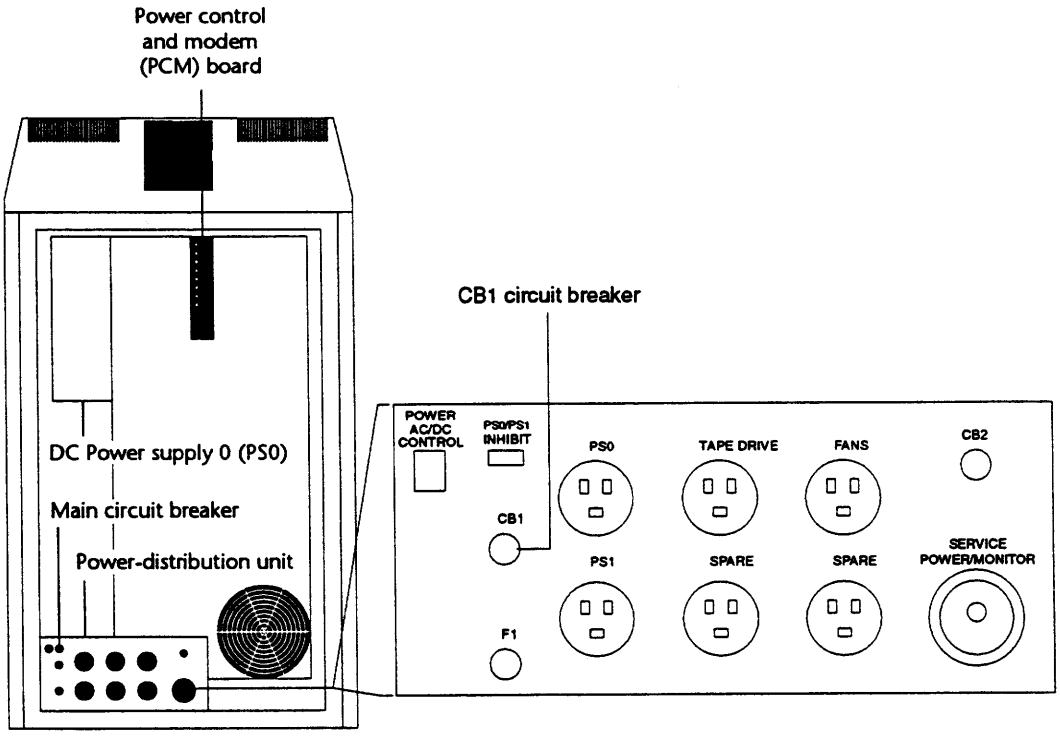

Figure 2-12 Series 300, CPU, Rear View: Location of Power-Distribution Unit and CB1 Circuit Breaker

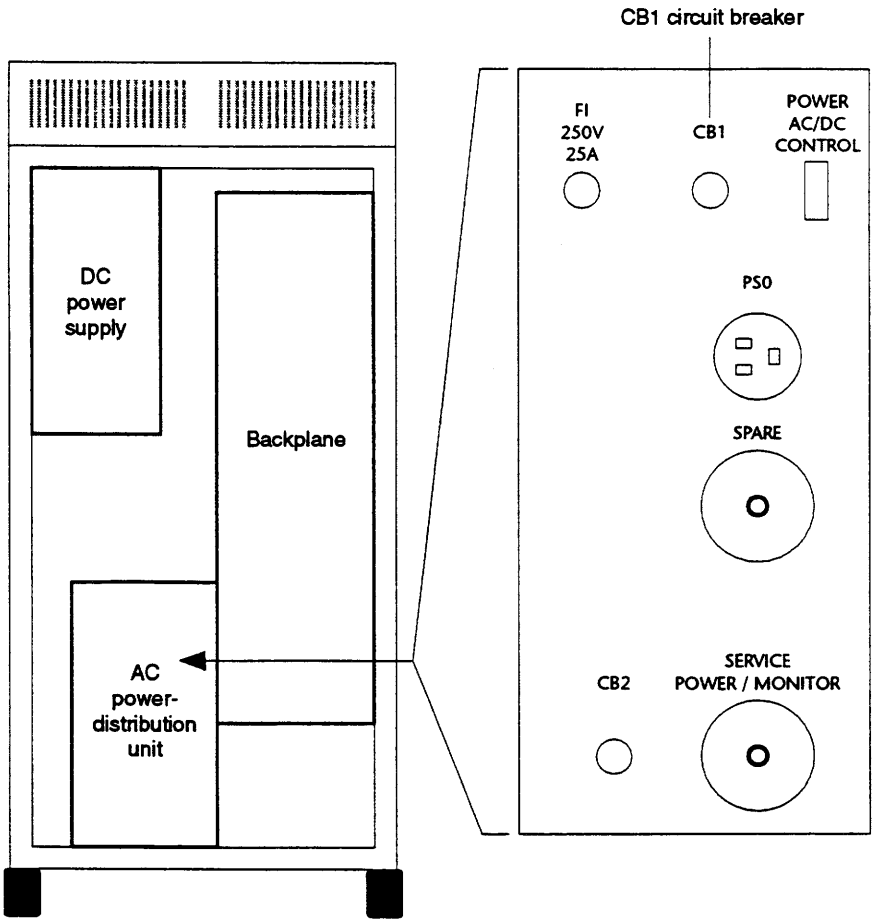

Figure 2-13 Series 400 Interior, Rear View: Location of Power-Distribution Unit and CB1 Circuit Breaker

DELIBERATE ENVIRONMENT CRASH USING THE OPERATOR'S CONSOLE [Break] KEY

The [Break] key halts the Environment abruptly.

Caution: *Use the* [Break] *key only if you cannot execute the Shutdown command from an Environment command window or from the* `command:` *prompt at the operator's console—for example, if the R1000 has hung. Do not use the* [Break] *key for scheduled Environment shutdowns; use the !Commands.Abbreviations-.Schedule_Shutdown command instead.*

The [Break] key is not recommended for normal Environment shutdown because:

- No shutdown warning is given to users.
- No snapshot is taken of the current Environment state. When the system reboots, it is restored to the state recorded at the last snapshot, and uncommitted work after that snapshot is lost.

Caution: *Do not use the* [Break] *key if the system has crashed, or important diagnostic information may be lost. Before using the* [Break] *key, contact the Rational Response Center about recording the Environment state before shutdown. (See Appendix C for information on recovering from a system crash.)*

To abruptly halt the Environment:

1. If possible, get to the console's `EEDB:` prompt (see Chapter 1).
2. From the console `EEDB:` prompt, run the snapshot command:

   ```
   Snapshot
   ```

3. Wait 15–30 seconds for the snapshot to complete.
4. Press the [Break] key.
5. A menu appears on the operator's console. If the operator-mode keyswitch is in the Interactive position and the system is a Series 400, the menu appears as:

   ```
   Please enter
       0 => Restart system
       1 => Ignore break key
       2 => Redisplay recent console output
       3 => Enter debugger
       4 => Reset tape scsi

   Enter option :
   ```

   If the system is a Series 200 or Series 300, option number 4 does not appear.

   For all systems, if the operator-mode keyswitch is in the Automatic position, the menu appears as:

   ```
   Please enter
      0 => Restart system
      1 => Ignore break key
      2 => Redisplay recent console output

   Enter option
   ```

   Table 2-1 describes each possible option.

Table 2-1 Description of [Break] Key Options

Option	Description
0	Causes an immediate Environment halt and reboot
1	Harmlessly exits the [Break] key menu without doing anything
2	Harmlessly exits the [Break] key menu and redisplays the last 256 characters from previous console messages or responses
3	Starts a low-level debugging session—choose this option only when instructed to do so by Rational support personnel
4	Resets the SCSI tape drive on Series 400 R1000s only

6. If you press [0] and then [Return], the following prompt appears:

   ```
   Do you really want to crash the system [N]?
   ```

7. Make the decision:
   - *To cancel* the request for a halt, press [Return]. The Environment state remains unchanged.
   - *To halt* the Environment, press [Y] and then [Return]. The Environment immediately halts and reboots.

   For information on the boot process, refer to the "Description of the Standard Boot Process" section, later in this chapter.

POWERING UP THE SYSTEM AFTER POWER-OFF

To restart the system after any power-off situation:

1. Ensure that the condition or failure that caused the shutdown is corrected.
2. Ensure that the operator-mode keyswitch on the CPU bay's control panel is turned to the Automatic position and that the power keyswitch is turned to the System Off or Standby position.
3. Verify that the main circuit breaker, CB1 on the power-distribution unit (PDU), is in the ON position (see Figure 2-11, 2-12, or 2-13).
4. Write-enable all the disk drives by switching each drive's write-protect switch to the OFF position (see Figure 2-4, 2-5, or 2-6).
5. Turn the power keyswitch to the System On position.

   The R1000 begins a series of power-on self-tests. As each self-test completes, the operator's console displays the result. The following is example output from the self-test.

```
R1000-200/300 IOC SELFTEST 6.0.0 ...
    512 KB memory ... [OK]
    Memory parity ... [OK]
    I/O bus control ... [OK]
    I/O bus map ... [OK]
    I/O bus map parity ... [OK]
    I/O bus transactions ... [OK]
    ATU ... [OK]
    ATU parity ... [OK]
    PIT ... [OK]
    Modem DUART channel ... [OK]
    Diagnostic DUART channel ... [OK]
    Clock / Calendar ... [OK]
    PCM devices ... [OK]
    Local interrupts ... [OK]
    ATU validity protection ... [OK]
    ATU write protection ... [OK]
    Illegal reference protection ... [OK]
    I/O bus parity ... [OK]
    I/O bus spurious interrupts ... [OK]
    Temperature sensors ... [OK]
    IOC diagnostic processor ... [OK]
    Power margining ... [OK]
    Clock margining ... [OK]
Selftest passed
```

   After successfully completing the self-tests, the R1000 begins to boot automatically. For more information, see "Description of the Standard Boot Process," below.

6. Occasionally the boot process started in step 5 above for the Series 200 or 300S cannot proceed because the disks are still accelerating (called *disk spinup*). In this case the operator's console displays the following menu:

```
The I/O Processor cannot be booted.  Please enter:
    0 => Try again
    1 => EPO system

Enter option : 0
```

To respond to this menu:

a. Wait 1 or 2 minutes to let the disks spin up.

b. Enter 0 and press [Return] to retry the boot.

Note: *Entering 1 turns the power off, which is equivalent to turning the power keyswitch to Standby.*

c. If the disks still are not ready and the same menu reappears, repeat steps 1 and 2.

d. The disks should take no more than 5 minutes to spin up. If the same menu appears after you have repeated steps 1 and 2, contact the Response Center (see Appendix D).

DESCRIPTION OF THE STANDARD BOOT PROCESS

The standard boot process starts the Environment after a power-off, a scheduled shutdown, or a [Break] key halt. The standard boot process applies when:

- Your system has not crashed because of a hardware or software failure, excessive temperature, or a power failure. (For information about booting your system after it has crashed, see Appendix C.)
- Your system's boot options have not been changed by your Rational support representative. (A customized boot process may vary slightly from the standard boot process for both the Automatic and the Interactive modes.)
- The operator-mode keyswitch on the R1000 system control panel is set to Automatic. If the operator-mode keyswitch is set to Interactive, the boot process may differ considerably from the standard boot process, especially if you respond to the question prompts with other than the suggested default values.

Interactive and Automatic Modes

If the operator-mode keyswitch is set to the Interactive position, the boot process pauses at the menu:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

Your Rational support representative can give you more information about the Interactive-mode options.

If the operator-mode keyswitch is set to the Automatic position, the standard boot process continues without pausing at this menu.

Note: *If the boot process pauses at any prompt or menu other than those described in this section, your system may have crashed. See Appendix C for information about diagnosing and recovering from a system crash.*

The Four Phases of the Boot Process

The standard system boot process is a four-phase chain of events. Understanding this chain of events helps you identify the messages and prompts that appear on the operator's console during the boot process. Knowing which messages and prompts are normal helps you to understand progress of the boot process and to identify any problems.

Once the boot process begins, it proceeds automatically; you interact with the system only once at the beginning of phase III. The following list summarizes the four phases of the boot process. More detailed descriptions of each phase follow.

- *Phase I:* The input-output controller (IOC) boots the input-output processor (IOP).
- *Phase II:* After the IOP boots, it brings up its own kernel and initializes the diagnostic file system (DFS).
- *Phase III:* The IOP prepares to boot the R1000 processor according to the specified subsystem configuration. (These subsystems are called the *kernel layer*, and the Standard configuration is the default.) The system then loads and initializes the kernel layer. The console displays messages as each subsystem executes. Once completed, the kernel layer provides an interface called the *R1000 kernel.* The console displays a title message similar to:

  ```
  ====>> Kernel.11.5.6 <<====
  ```

  and the console prompt looks like:

  ```
  Kernel:
  ```

  The R1000 restarts virtual memory and continues initializing subsystems until one called the *Environment elaborator database* (EEDB) is loaded.
- *Phase IV:* After the EEDB is loaded, the console displays a title message similar to:

  ```
  ====>> Elaborator Database <<====
  ```

  and the console prompt looks like:

  ```
  EEDB:
  ```

  The EEDB initializes a second set of subsystems, called the *Environment layer.* The console displays messages as each subsystem executes.

  At the end of phase IV, the EEDB executes initialization procedures contained in an Environment world called !Machine.Initialization. (For details about machine initialization, see Appendix E.)

  When the boot process is finished, the operator's console and all connected Environment terminals display the following prompt:

  ```
  username:
  ```

The following subsections include *sample* messages similar to those displayed during a normal boot. For brevity, most of the boot messages are omitted. The listed messages either provide key information or request operator input.

Phase I

When the boot process starts, dialogue similar to the following is displayed:

```
R1000-200/300 IOC SELFTEST 6.0.0 ...
    512 KB memory ... [OK]
    Memory parity ... [OK]
    I/O bus control ... [OK]
    I/O bus map ... [OK]
    I/O bus map parity ... [OK]
    I/O bus transactions ... [OK]
    ATU ... [OK]
    ATU parity ... [OK]
    PIT ... [OK]
    Modem DUART channel ... [OK]
    Diagnostic DUART channel ... [OK]
    Clock / Calendar ... [OK]
    PCM devices ... [OK]
    Local interrupts ... [OK]
    ATU validity protection ... [OK]
    ATU write protection ... [OK]
    Illegal reference protection ... [OK]
    I/O bus parity ... [OK]
    I/O bus spurious interrupts ... [OK]
    Temperature sensors ... [OK]
    IOC diagnostic processor ... [OK]
    Power margining ... [OK]
    Clock margining ... [OK]
Selftest passed

Restarting R1000-200 August 14th, 1992 at 11:23:40

Logical tape drive 0 is a 8MM cartridge tape drive.
Logical tape drive 1 is declared non-existent.
Logical tape drive 2 is declared non-existent.
Logical tape drive 3 is a 1/2 inch 9 track tape drive at physical
unit 0.

Booting I/O Processor
IOC Bootstrap Version 5.7
```

After all the disks have spun up, the boot process continues with phase II.

Phase II

After the IOP boots, it brings up its own kernel and initializes the diagnostic file system.

The following messages are displayed:

```
Initializing I/O Processor Kernel n_n_n
Disk Controller 0, Disk  0 is ONLINE and WRITE ENABLED
Disk Controller 0, Disk  1 is ONLINE and WRITE ENABLED
Disk Controller 0, Disk  2 is ONLINE and WRITE ENABLED
Disk Controller 0, Disk  3 is ONLINE and WRITE ENABLED
```

```
IOP Kernel is initialized
Initializing diagnostic file system ... [OK]

   FREEZE_WORLD.FIU
   MF.IOC
   MF.VAL
   MF.TYP
   RESET.SEQ
   MF.MEM
   MF.MEM
   MF.MEM
   MF.MEM
   MF.FIU

================================================
```

where *n_n_n* refers to the current IOP kernel version number.

Phase III

The IOP prepares to boot the R1000 processor. The IOP needs to know what configuration of subsystems to use. With some configurations, the system pauses at the following menu:

```
Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

Note: *Some custom systems display a different menu or do not pause at this point.*

If your system displays this or a similar menu, press [Return] when you want the boot process to continue. Pressing [Return] automatically specifies the default configuration, which is contained in a file called Standard. You may never need to specify a configuration file other than Standard.

From this point on, the boot process requires no further input from you.

The system next reads the subsystems that form the standard configuration:

```
--- Booting the R1000 Environment ---
   READ_NOVRAM_DATA.TYP
   READ_NOVRAM_DATA.VAL
   READ_NOVRAM_DATA.FIU
   READ_NOVRAM_DATA.SEQ
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.SEQ
   READ_NOVRAM_DATA.FIU
   READ_NOVRAM_DATA.TYP
   READ_NOVRAM_DATA.VAL
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
   READ_NOVRAM_DATA.MEM
```

```
   LOAD_CONFIG.MEM
   CLEAR_TAGSTORE.MEM
   CLEAR_PARITY_ERRORS.MEM
   LOAD_CONFIG.MEM
   CLEAR_TAGSTORE.MEM
   CLEAR_PARITY_ERRORS.MEM
   LOAD_CONFIG.MEM
   CLEAR_TAGSTORE.MEM
   CLEAR_PARITY_ERRORS.MEM
   LOAD_CONFIG.MEM
   CLEAR_TAGSTORE.MEM
   CLEAR_PARITY_ERRORS.MEM
   CLEAR_PARITY.FIU
   CLEAR_PARITY.VAL
   CLEAR_PARITY.TYP
   CLEAR_PARITY.SEQ
   File  : M207_17.M200_UCODE
Bound : August 8, 1992 at 11:45:04 AM
Delta : !R1000.MICROCODE.M200_WORKING.UNITS.UCODE_DB
Mom   :
Loading Register Files .... [OK]
   LOAD_BENIGN_UWORD.TYP
   SET_HIT.MEM
   SET_HIT.MEM
   SET_HIT.MEM
   SET_HIT.MEM
   INIT_MRU.FIU
   CLEAR_HITS.MEM
   CLEAR_HITS.MEM
   CLEAR_HITS.MEM
   CLEAR_HITS.MEM
   PREP_RUN.TYP
   PREP_RUN.VAL
   PREP_RUN.IOC
   PREP_RUN.SEQ
   PREP_RUN.FIU
   CLEAR_PARITY_ERRORS.MEM
   CLEAR_PARITY_ERRORS.MEM
   CLEAR_PARITY_ERRORS.MEM
   CLEAR_PARITY_ERRORS.MEM
   FREEZE_WORLD.FIU
   RUN_NORMAL.TYP
   RUN_NORMAL.VAL
   RUN_CHECK.SEQ
   RUN_CHECK.IOC
   RUN_CHECK.MEM
   RUN_CHECK.MEM
   RUN_CHECK.MEM
   RUN_CHECK.MEM
   RUN_NORMAL.FIU
Loading : KAB.11.0.0.MLOAD
Loading : KMI.11.0.0.MLOAD
Loading : KKDIO.11.0.3.MLOAD
Loading : KKD.11.0.0S.MLOAD
Loading : KK.11.5.5K.MLOAD
Loading : EEDB.11.1.3D.MLOAD
Loading : UOSU.11.3.0D.MLOAD
Loading : UED.10.0.0R.MLOAD
```

```
Loading : UM.11.1.5D.MLOAD
Loading : UAT.11.2.2D.MLOAD
854/1532 wired/total pages loaded.

The use of this system is subject to the software license terms and
    conditions agreed upon between Rational and the Customer.

                    Copyright 1992 by Rational.

                     RESTRICTED RIGHTS LEGEND

    Use, duplication, or disclosure by the Government is subject to
    restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in
    Technical Data and Computer Software clause at DoD FAR Supplement
    252.227-7013.

            Rational
            3320 Scott Blvd.
            Santa Clara, California 95054-3197
```

Next, the system restarts virtual memory and then produces an interface called the *R1000 kernel*, which initializes various subsystems.

```
====>> Kernel.11.5.5 <<====
Kernel: CHANGE_GHOST_LOGGING
WANT TRACING: FALSE
WANT LOGGING: FALSE
Kernel: START_VIRTUAL_MEMORY
ALLOW PAGE FAULTS: YES

====>> CONFIGURATOR <<====
starting diagnosis of configuration
starting virtual memory system

====>> ERROR_LOG <<====
14:54:02 --- ENP_Driver.Worker.Finalized

====>> CONFIGURATOR <<====
the virtual memory system is up

====>> Kernel.11.5.5 <<====
Kernel: START_NETWORK_IO
Kernel: START_ENVIRONMENT
TRACE LEVEL: INFORMATIVE
Kernel:
====>> Environment Elaborator <<====
Elaborating subsystem:  ENVIRONMENT_DEBUGGER
Elaborating subsystem:  ABSTRACT_TYPES
Elaborating subsystem:  MISCELLANEOUS
Elaborating subsystem:  OS_UTILITIES
Elaborating subsystem:  ELABORATOR_DATABASE
```

The R1000 continues to initialize subsystems until it has initialized one called the *Environment elaborator database* (EEDB).

Note: *During the boot process, some kernel prompts appear to be blank and waiting for user input. They are not waiting for anything. The machine is quite busy; let it boot on its own.*

Phase IV

After the EEDB is loaded, it initializes a second set of subsystems, called the *Environment layer*. Typical console messages during this process are:

```
====>> Kernel.11.5.5 <<====
Kernel: Elaborator Database <<=====
NETWORK.11.1.3D
OM_MECHANISMS.11.1.5D

====>> Kernel.11.5.5 <<====
Kernel:
BASIC_MANAGERS.11.2.5D
ADA_MANAGEMENT.11.50.3D
DISK_CLEANER.11.1.2D
PARSER.11.50.0D
PRETTY_PRINTER.11.50.1D
DIRECTORY.11.4.2D
INPUT_OUTPUT.11.6.2D
COMPILER_UTILITIES.11.50.1D
SEMANTICS.11.50.1D
R1000_DEPENDENT.11.50.0D
R1000_CHECKING.11.50.1D
R1000_CODE_GEN.11.50.3D
IMAGE.11.4.1D
CORE_EDITOR.11.6.0D
TOOLS.11.5.0D
OE_MECHANISMS.11.1.1D
OBJECT_EDITOR.11.5.1D
MAIL.DELTA
OS_COMMANDS.11.6.0D
COMMANDS.11.5.1D
FTP_INTERFACE.11.1.0D
TOOLS_INTEGRATION.DELTA
CMVC.11.7.0D
DESIGN_FACILITY.DELTA
ARCHIVE.11.3.6D
NATIVE_DEBUGGER.11.1.6D
CROSS_DEVELOPMENT.DELTA
INITIALIZE.11.2.3D
```

After the Environment layer is loaded, the console displays the following prompt:

```
EEDB:
```

This prompt reappears several times during phase IV. No input is required.

At the end of phase IV, the EEDB executes machine-initialization procedures that are defined in the world !Machine.Initialization and terminals are enabled for login. For more information about machine initialization, see Appendix E.

During machine initialization, time-stamped status messages are sent to both the console and the error log.

```
====>> Elaborator Database <<====
15:20:11 +++ Operator Enable_Terminal  16
15:22:19 +++ Operator Enable_Terminal  16
15:22:19 +++ Operator Enable_Terminal  235
15:22:21 +++ Operator Enable_Terminal  236
15:22:21 +++ Operator Enable_Terminal  237
```

```
15:22:21 +++ Operator Enable_Terminal  238
15:22:22 +++ Operator Enable_Terminal  239
15:22:23 +++ Operator Enable_Terminal  240
15:22:23 +++ Operator Enable_Terminal  241
15:22:24 +++ Operator Enable_Terminal  242
15:22:25 +++ Operator Enable_Terminal  243
15:22:26 +++ Operator Enable_Terminal  244
15:22:26 +++ Operator Enable_Terminal  245
15:22:27 +++ Operator Enable_Terminal  246
15:22:27 +++ Operator Enable_Terminal  247
15:22:28 +++ Operator Enable_Terminal  248
15:22:28 +++ Operator Enable_Terminal  249
```

At the end of the boot process, the following two lines appear on the operator's console:

```
====>> Console Command Interpreter (System Job 183) <<====
username:
```

When the `username:` prompt appears, phase IV is complete.

(Chapter 2 continues after the following red tab.)

EMERGENCY POWER-OFF

Caution: *Do* ***not*** *proceed with the following emergency procedure in a non-emergency situation: use the procedures in the "Normal Power-Off" section earlier in this chapter.*

For emergencies only:

1. *Optional snapshot:* If you do not have time to save the Environment state or cannot get to the console's `EEDB:` prompt, skip to step 2. Otherwise, take a snapshot:
   a. From the console's `EEDB:` prompt, enter the `snapshot` command.
   b. Wait 15–30 seconds until the console message "snapshot is finished" appears.
2. Remove power from the R1000.
   - Series 200: Locate the power keyswitch, insert the key, and turn the key to the System Off position (see Figure 2-14).
   - Series 300 or 400: Locate and press the Emergency Off button (see Figure 2-15 or 2-16).

   This removes all AC and DC power from the R1000, except AC power to the power-distribution unit (PDU) and, on the Series 200 and 300, the cooling fans.
3. If you need to remove *all* power from the R1000, open the rear cabinet door and switch the circuit breaker switch (CB1) to the OFF position. See Figure 2-17, 2-18, or 2-19.

   Alternatively, you can unplug the power cord or trip the main circuit breaker supplying line power to the R1000.

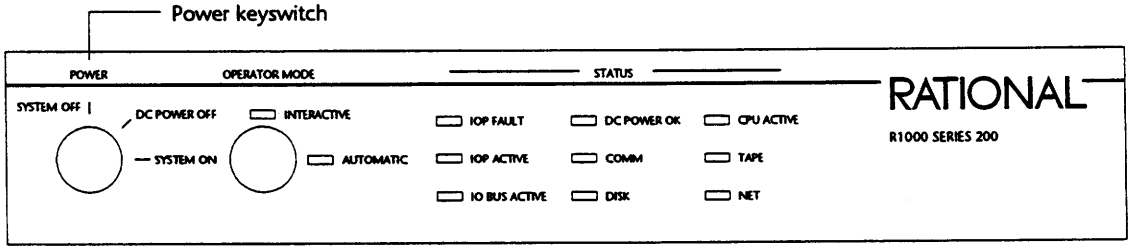

Figure 2-14 Series 200 Control Panel: Power Keyswitch Used for Emergency Power-Off

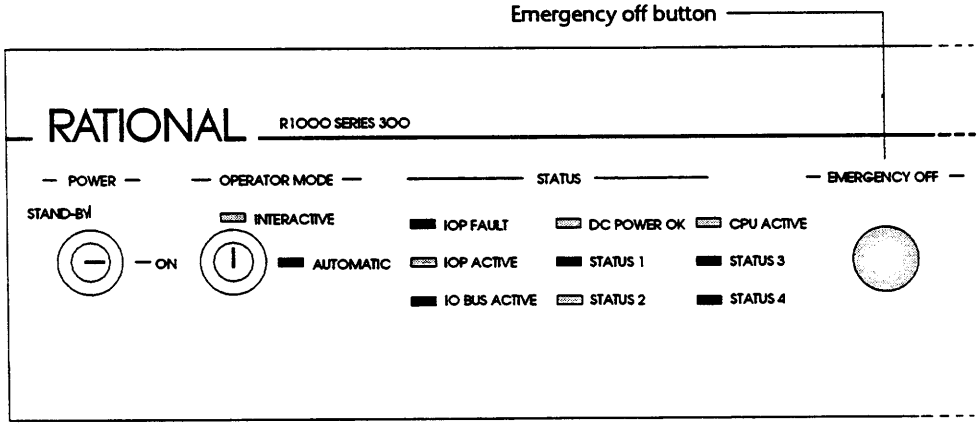

Figure 2-15 Series 300 Control Panel: Emergency Off Button

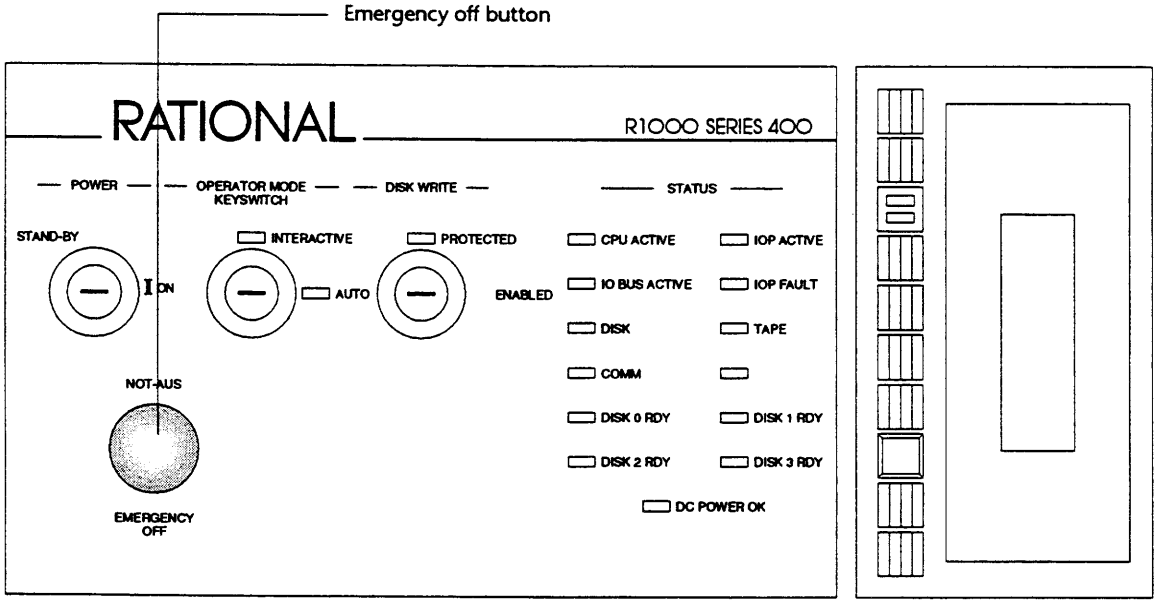

Figure 2-16 Series 400 Control Panel: Emergency Off Button

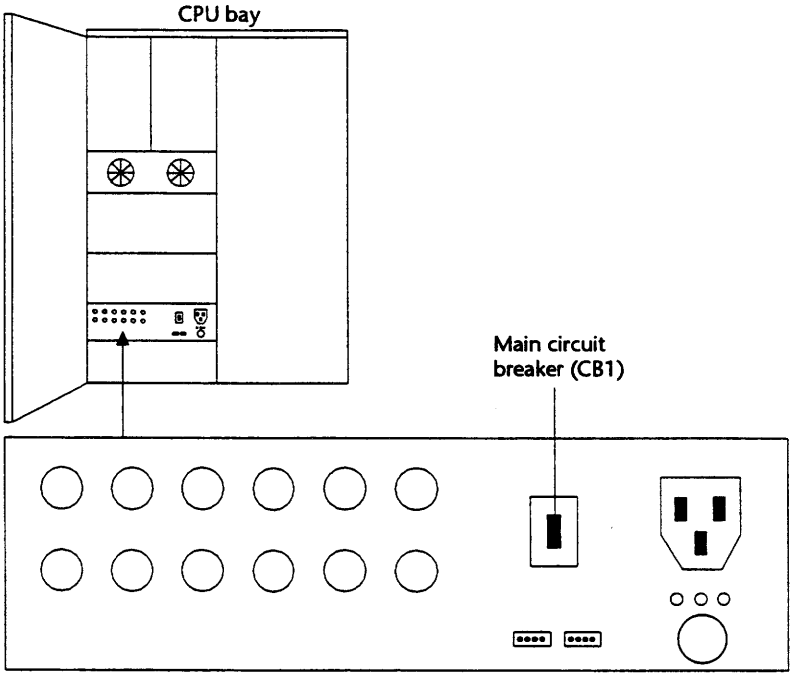

Figure 2-17 Series 200: Location of CB1 Circuit Breaker

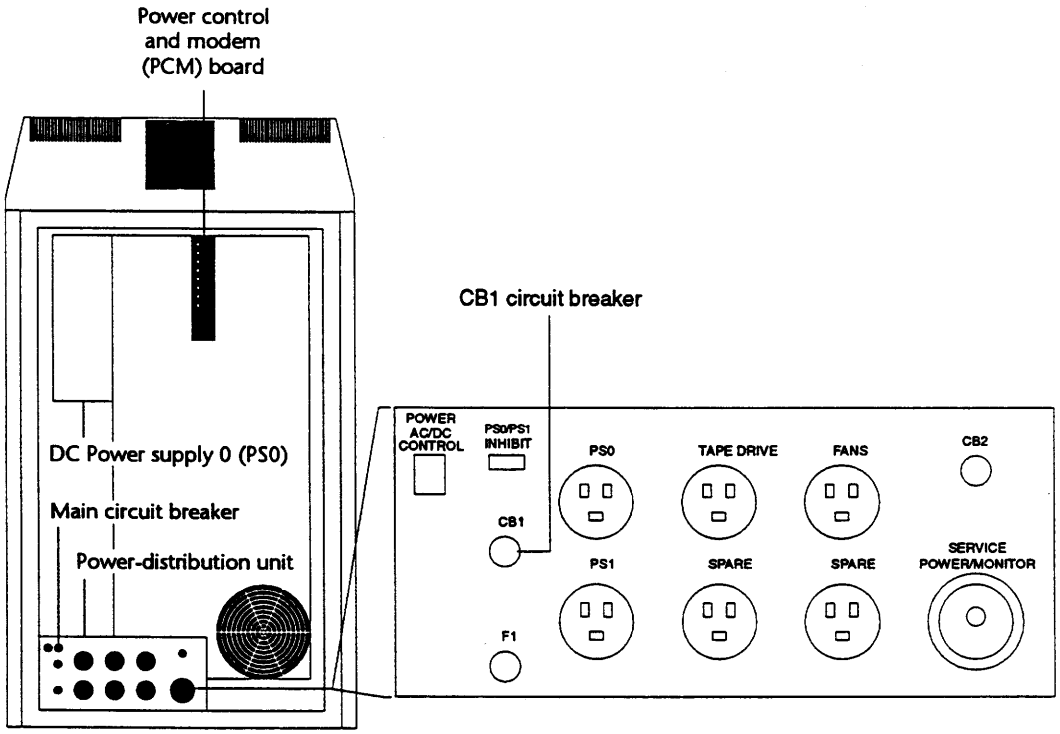

Figure 2-18 Series 300, CPU, Rear View: Location of Power-Distribution Unit and CB1 Circuit Breaker

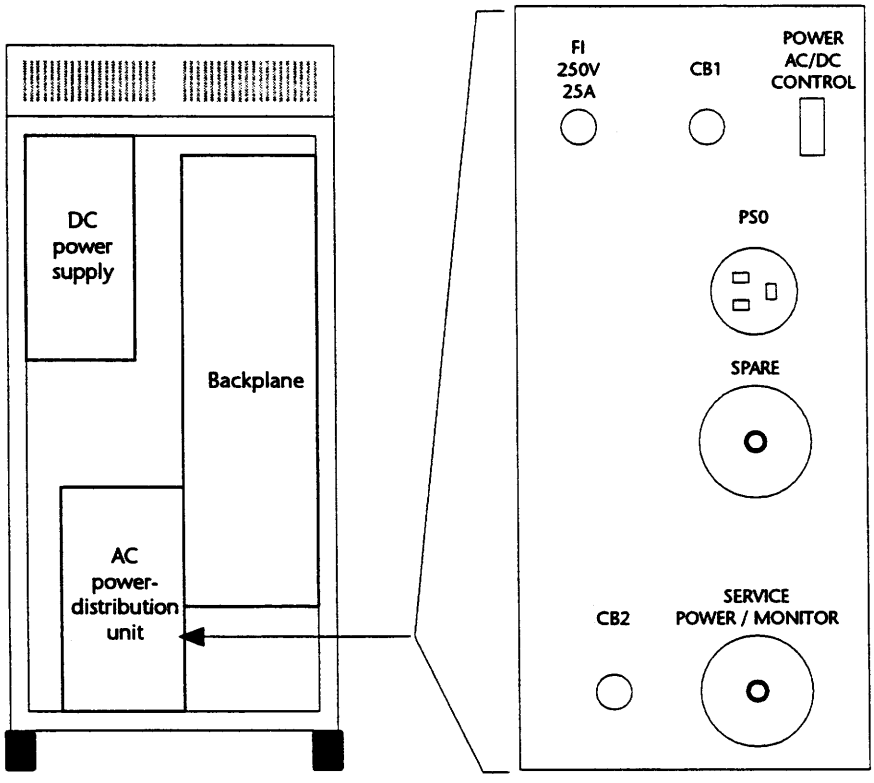

Figure 2-19 Series 400 Interior, Rear View: Location of Power-Distribution Unit and CB1 Circuit Breaker

3

Managing User Accounts

To log into the Environment, each user must have access to a user account that identifies a user to the Environment through a *username* and a *password*. Each user account reserves a world in the library hierarchy in which a user can collect his or her work. Finally, users can tailor aspects of the Environment to suit their needs by creating specialized sessions within their user accounts.

This chapter describes the components of a user account and how to create, disable, and reactivate user accounts, change passwords, log out a user, monitor account activity, and customize account defaults.

COMPONENTS OF A USER ACCOUNT

The major components of a user account include:

- A username
- A password
- A home world (library)
- One or more sessions
- A searchlist
- Session switches (optional)
- A customized login procedure (optional)

Username and Password

Together, the username and password identify a user to the Environment and enable the user to log into the account. Whereas usernames typically are known to other Environment users, an account password should be known only to the owner of the account. When a user logs in, the Environment compares the username and password pair to ensure system security.

Usernames must be legal Ada simple names. A list of current usernames is maintained automatically in !Machine.Users.

Home World

Each user account reserves a world in the Environment library hierarchy in which a user can collect and organize his or her work. The world associated with a user account is called the user's *home world*. All home worlds reside in !Users.

An account's home world takes its name from the account username. For example, if an account's username is Anderson, the name of the home world for that account is !Users.Anderson.

The user's home world is displayed whenever the user presses the [Home Library] key or executes the !Commands.What.Home_Library command.

Sessions

Each user account has associated with it a default session (called S_1), plus any additional sessions the user has created. Sessions can be used to tailor characteristics of the Environment so that it best supports a particular user's working requirements. For example, if a user is part of two projects, Project 1 and Project 2, this user can create two sessions, giving them appropriate names, such as Proj_1 and Proj_2. The user then can customize the login, searchlist, key bindings, switches, and activities for each of these sessions to match each project's unique characteristics. For more information about sessions, see the Session and Job Management (SJM) and Project Management (PM) books of the *Rational Environment Reference Manual.*

A user can have any number of sessions, although the user can log into only one at a time on a given port. (Users who have access to multiple ports, however, can log into multiple sessions.) To log into a particular session, specify the session name at the `session:` prompt during login. If no name is specified at the session prompt, the default session S_1 is used.

Login Procedures

Each time a user logs in, the Environment executes a login procedure. For new accounts, the systemwide default login procedure, !Machine.Release.Current.Commands.Login, is executed. The default login procedure contains Environment commands that display the user's home world, open a command window, set up the default activity, and display the daily message. You can tailor the systemwide default login to contain procedures that are relevant to all users at your installation (see the "Default Login Procedure" subsection, later in this chapter, for more information about customizing the default login procedure).

Any user can override the default login procedure by creating a customized login procedure in the home world of his or her account. For example, an individual user's login procedure might be used to customize which windows automatically appear on the screen at login. If an account contains a customized login procedure, the default login procedure is not executed. Therefore, if a user wants his or her home world displayed, a command window opened, the default activity established, and the daily message displayed, the user must include the appropriate commands in the customized login procedure.

Customized login procedures must be called Login and are usually created in the user's home account. For example, a user whose username is Anderson might have a customized login procedure called !Users.Anderson.Login. However, a customized login procedure can be located anywhere as long as the user's searchlist contains an entry that resolves to the customized login.

You might also want to implement a systemwide login procedure that can*not* be overridden. To do so, add a coded, parameterless procedure called System_Login

to !Machine. This procedure will be executed unconditionally, before either the user's customized login procedure or the default login procedure is run. The unconditional execution of System_Login is useful for sending vital messages, enhancing security, and doing other operations you do not want users to override. See the "Systemwide Login Procedure" subsection, later in this chapter, for more information about this option.

The order of operations of these login procedures is summarized below:

- If the System_Login procedure has been implemented, it is run first.
- If the user has a customized login, it is run; otherwise, the system default login is run.

Special Accounts

The Environment always contains a special, privileged account with the username Operator. Like other accounts, this account has a home world (!Users.Operator) and a default session (S_1). The Operator account is created automatically the first time the system boots. If the account is ever disabled, it is recreated automatically the next time the system boots. Refer to the "Operator" subsection in Chapter 4 for more information about this special account.

A second special account has the username Rational, a home world (!Users-.Rational), and a default session (S_1). This account is for Rational support personnel to use when maintaining your system.

Finally, a third special account called *System is used by the system to run system jobs. This account is unusual, because it has no home world or session.

CREATING USER ACCOUNTS

Before you can disable a user account, you must be logged into the Environment as Operator.

New user accounts are created using the !Commands.Operator.Create_User procedure.

To create a new user account:

1. Log into the Environment as Operator.
2. Enter the following in a command window:

```
Operator.Create_User
```

3. Press [Complete].

   The system responds:

```
Operator.Create_User (User     => ">>USER NAME<<",
                      Password => "",
                      Volume   => 0,
                      Response => "<PROFILE>");
```

4. Fill in the parameters as follows:
   - User: Specifies the username for the account. The name must be unique and must be a legal Ada simple name.
   - Password: Specifies a password for the user's first login. A password can be any arbitrary string, including the null string (unless password policies require it to be at least *n* characters long).

- Volume: Specifies the volume on which to create the user's home world. If you leave the value as 0 (the default), the Environment chooses the volume with the most available space.
- Response: Specifies the system response in case of errors. The default specifies the job response profile. (For more information, see the "Parameter-Value Conventions" section of the Reference Summary (RS) book of the *Rational Environment Reference Manual.*)

  For example:

```
Operator.Create_User (User     => "Anderson",
                      Password => "Andersonnew",
                      Volume   => 2,
                      Response => "<PROFILE>");
```

5. Press [Promote].
6. Advise the user about how to change the temporary password to a more secure password (see "Changing Passwords," below).

Executing the sample command shown above creates the following:

- The user Anderson with a password of Andersonnew.
- An entry for Anderson in the world !Machine.Users, which lists all usernames.
- The world !Users.Anderson on volume 2 (unless a world with this name already exists; see "Reactivating User Accounts," below).
- A default session called S_1 for the account. S_1 automatically uses the default searchlist stored in the !Machine.Search_Lists.Default file.
- A group Anderson, which is added to groups Public and Network_Public.

See Chapter 4 for information on the ACLs for new user accounts.

Like most commands, the !Commands.Operator.Create_User command can be entered from the operator's console command interpreter.

DISABLING USER ACCOUNTS

Before you can disable a user account, you must be logged into the Environment as Operator.

To disable a user account:

1. Make sure that no one is logged into the account. To do this, enter and promote the following command from a command window or from the operator's console command interpreter (see "Displaying Current Login Information," below):

   ```
   What.Users
   ```

   When the display appears, search for the account's username. If you find the username in the display, tell the user to log off, or use Operator.Force_Logoff (see "Logging Off a User" section, below).
2. Enter the following command in a command window:

   ```
   Operator.Delete_User
   ```
3. Press [Complete].

   The system responds:

```
Operator.Delete_User (User     => ">>USER NAME<<",
                      Response => "<PROFILE>");
```

4. Fill in the parameters as follows:
   - User: Specifies the username for the account you want to disable.
   - Response: Specifies the system response in case of errors. The default specifies the job response profile.

   For example:

```
Operator.Delete_User (User     => "Anderson",
                      Response => "<PROFILE>");
```

5. Press [Promote].

Executing this command produces the following results:

- The username Anderson is removed from the world !Machine.Users, disabling logins to the account.
- The default session S_1 is deleted.
- The world !Users.Anderson is preserved, even though it no longer serves as the home world of a login account. At this point, you can keep !Users.Anderson if the world contains work that will be needed after Anderson is gone or if you intend to reactivate the account at a later time (as described below). Or, you can archive world !Users.Anderson to tape and then delete it using the !Commands-.Library.Destroy command or CMVC commands as appropriate.
- Anderson is removed from all groups of which it was a member. Group Anderson is deleted.

Note: *Do not disable the Operator account, because the Operator's password is required for various procedures. A disabled Operator account can be reactivated by rebooting the system and then resetting the password or, if you have operator capability, by following the procedure described in "Reactivating User Accounts," below.*

Like most commands, the !Commands.Operator.Delete_User command can be entered from the operator's console command interpreter.

REACTIVATING USER ACCOUNTS

Occasionally, you may need to reactivate a user's account after disabling it. If the user's home world was not deleted after the account was disabled, then reactivating the account allows logging into the original home world.

To reactivate a user account after you have disabled it, follow the same steps as for creating a new account. Enter the !Commands.Operator.Create_User procedure, specifying the username and temporary password. If the home world for the username already exists, the Volume parameter is ignored. For example:

```
Operator.Create_User (User     => "Anderson",
                      Password => "Temp",
                      Volume   => 0,
                      Response => "<PROFILE>");
```

Like most commands, the !Commands.Operator.Create_User command can be entered from the operator-console command interpreter.

If you use !Commands.Archive.Restore to restore a currently nonexistent user's home world, the user is created automatically (with the same password the user had when the account was archived).

PASSWORD POLICY

See the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, package Operator, for a complete discussion and procedures concerning:

- Setting password policy and change schedules
- Forgotten operator password
- Password expirations

CHANGING PASSWORDS

To change the password for an account:

1. Enter the following command in a command window:

   ```
   Operator.Change_Password
   ```

2. Press [Complete].

   The system responds:

   ```
   Operator.Change_Password (User         => ">>USER NAME<<",
                             Old_Password => "",
                             New_Password => "",
                             Response     => "<PROFILE>");
   ```

3. Fill in the parameters as follows:
   - User: Specifies the username of the account.
   - Old_Password: Specifies the current password for the account. If this password is not known, the Operator's password can be used.
   - New_Password: Specifies the new password for the account.
   - Response: Specifies the system response in case of errors. The default specifies the job response profile.

     For example:

     ```
     Operator.Change_Password (User         => "Anderson",
                               Old_Password => "Temp",
                               New_Password => "Hello",
                               Response     => "<PROFILE>");
     ```

4. Press [Promote].

Note that anyone can change the password for any account if the old password—or the password for the Operator's account—is known.

Like most commands, the !Commands.Operator.Change_Password command can be entered from the operator's console command interpreter.

Initializing the Operator's Account Password

When the Operator's account is created, it has the null string ("") as its password (unless a password policy is in effect that requires more than 0 characters). It is recommended that you change this initial password to a more secure password as soon as possible. The Operator's account password is important because you cannot delete user accounts or change forgotten passwords without it.

For example, to change the initial null-string password to the string "secret", execute the following command in a command window:

```
Operator.Change_Password (User         => "Operator",
                          Old_Password => ""
                          New_Password => "Secret",
                          Response     => "<PROFILE>");
```

Note: *If the password to the Operator's account has been forgotten, it can be reset by another job running in privileged mode (see the "Privileged" subsection in Chapter 7).*

Like most commands, the !Commands.Operator.Change_Password command can be entered from the operator's console command interpreter.

Expiration of Operator Password

A special accommodation is made for Operators who did not change their passwords before they expired. If the Operator tries to log in with an expired password (and no current password exists), a message to this effect is displayed and a login prompt appears requesting an authorization code. The Operator can obtain a unique authorization code from Rational customer support and enter this code at the appropriate prompt. Note that:

- The Operator must know the old, expired password.
- The authorization code is valid only for a specific machine on the current date.
- The authorization code is case-sensitive (all uppercase).

See the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, package Operator, for more information.

LOGGING OFF A USER

A user normally logs off using the Editor.Quit procedure from his or her own terminal. Under certain circumstances, you may need to log a user off from a different terminal. For example, if a user logs into the Environment on a terminal whose type does not match its port setting, the Environment interprets the character sequences from the terminal incorrectly. As a result, the user will not be able to log off using Quit from that terminal. Either the user can press [Break] three times or the session can be terminated from another terminal.

To terminate a session from another terminal:

1. Log into the Environment as Operator.
2. Find the port number of the user's terminal. To do this, run the following command in a command window or from the operator's console command interpreter (see "Displaying Current Login Information," below):

   ```
   What.Users;
   ```

   When the display appears:

   a. Locate the username in the User column.

   b. If the user is logged into more than one session, locate the appropriate session name in the Job Name column.

c. Finally, note the port number that appears in the Line column of the same line as in step b.

3. Log off the user's session by entering and promoting the following command (*n* is the port number noted in the preceding step; the Commit_Buffers parameter specifies whether the user's uncommitted changes should be saved):

```
Operator.Force_Logoff
        (Physical_Line => n, Commit_Buffers => True);
```

The session active on the specified line is terminated. Any uncommitted changes to images are saved. The user's background jobs (if any) continue to run, and any foreground jobs that do not require interactive input are put in the background.

Foreground jobs that do attempt interactive input or output will raise !Io.Io_Exceptions.End_Error. You may want to kill these jobs with the Job.Kill command before forcing off the user.

Like most commands, the What.Users and Operator.Force_Logoff commands can be entered from the operator's console command interpreter.

MONITORING ACCOUNT ACTIVITY

The Environment provides several ways to monitor account activity:

- You can execute the What.Users command to display current account activity.
- You can inspect the accounting files in !Machine.Accounting to obtain a summary of all account activity that occurred from one system boot to the next.
- You can use the "Job Accounting" tool described in the *Rational Software Library Catalog* to display the contents of !Machine.Accounting files.

Displaying Current Login Information

With the What.Users command, you can display current login information such as:

- A list of all users currently logged in
- How long each user has been logged in
- The port number on which each user is logged in
- A list of sessions and jobs started by each user

Executing the What.Users command is especially useful when you want to make sure a particular user is not logged in (for example, so you can use the Operator.Delete_User command) or when you want to find out which port a user's session is on (for example, so you can use the Operator.Force_Logoff command).

To display current login information:

1. Execute the following in a command window or from the operator's console command interpreter:

```
What.Users;
```

2. This command returns a display of user activity at one point in time. To update the display, reenter the command (the display is static, unlike the What.Jobs command).

The What.Users command produces a display such as the following in the standard output window:

```
User Status on August 20, 1992 at 04:22:52 AM

  User     Line  Job   S      Time          Job Name
========  ====  ====  ====  ========  ========================
*SYSTEM      -     4  RUN   22/19:04  System
                   5  RUN   22/19:04  Daemons
                 249  IDLE  22/19:01  Ftp Server

DHE         26   223  IDLE  02/15:21  [DHE.S_1 Command]
ANDERSON    19   166  RUN   06:43:26  [ANDERSON.S_1 Command]
                 230  RUN   00:03.39  !COMMANDS.OPERATOR'V(1)V
                                       % WHAT.USERS
WET         21   158  RUN   53:04:96  [WET.TEST Command]
OPERATOR    16   202  RUN   08:17:06  [OPERATOR.S_1 Command]
ANDERSON    30   244  RUN   02:14:32  [ANDERSON.DOCS Command])
```

The headings are described below:

- User: The username of the account. If more than one job belongs to a single user, the username is shown only for the first of those jobs. Entries for subsequent jobs belonging to the same user appear with blanks in the User column.
- Line: The port number on which the user is logged in. A dash (-) in the Line column means that the user is not logged into the Environment but has one or more jobs running.
- Job: The number of a job the user has started.
- S: The status of a user job or session. See the What.Users procedure in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual* for a description of job statuses.
- Time: The elapsed time since the job or session began.
  - *mm:ss.fff* indicates the number of minutes, seconds, and decimal fractions of seconds of elapsed time. In the above example, the job What.Users has run for a little more than 3 seconds, or 00:03.39.
  - *hh:mm:ss* indicates the number of hours, minutes, and seconds of elapsed time. In the above example, the Operator has been logged in for 8 hours, 17 minutes, and 6 seconds, or 08:17:06.
  - *dd/hh:mm* indicates the number of days, hours, and minutes of elapsed time. In the above example, user DHE has been logged in for more than 2.5 days, or 2/15:21.
- Job Name: The name of the user's session or the name of a job the user has started. Note that one username (in the above example, Anderson) can have more than one session (S_1 and DOCS). Also note that session names are not necessarily grouped together by usernames, whereas jobs are grouped together by session name.

The entry for *SYSTEM indicates how long the system has been up. Jobs listed under *SYSTEM were started by !Machine.Initialize.

Obtaining Accounting Information

Some installations require that users or user groups be billed for the system resources that they use. To help you account for resource use, you can request that the Environment collect accounting information, which includes:

- The login and logout time for each user session
- The number of jobs that were started and completed during each session
- The elapsed CPU time and the total number of disk requests for all jobs that were started and completed during each session
- The start and finish time for each system job and for each background user job that completed after the user logged out
- The elapsed CPU time and the total number of disk requests for all system jobs and background user jobs that completed after the user logged out

The Environment will not collect accounting information unless you create a world called Accounting.Enabled in the world !Machine. Once you have created this !Machine.Accounting.Enabled world, the Environment automatically creates a file for accounting information each time the system boots. Information is collected in this file until the system shuts down. When the system boots again, a new file is created. Therefore, each file contains a summary of the account activity from system boot to system boot, so, depending on how regularly the system is booted, some files may be quite short and others may be quite long.

The name for each file in !Machine.Accounting contains the date and time it was created (and, consequently, the date and time of each system boot). For example, the world !Machine.Accounting might contain entries such as:

```
Activity_92_08_07_At_09_32_34
Activity_92_08_11_At_16_31_31
Activity_92_08_22_At_09_15_27
Activity_92_08_29_At_09_27_14
```

The file Activity_92_08_11_At_16_31_31 contains accounting information for the period of time between August 11, 1992, at 4:31 P.M. and August 22, 1992, at 9:15 A.M. (the time the next file was created).

Following are some sample entries from the file Activity_92_08_11_At_16_31_31:

```
S 08/11/92 19:50:06 08/11/92 19:51:58  00:12.435  200  10 WEST.S_1
S 08/11/92 17:24:47 08/12/92 09:14:54  03:39.558 2562  62 DHE.S_1
S 08/12/92 09:09:58 08/12/92 12:20:59  22:13.833 7071 624 DCG.TEST
J 08/12/92 09:39:52 08/12/92 12:25:00  01:19.840 1209   1 DCG.TEST
S 08/12/92 06:49:28 08/12/92 22:01:16  01:05.013 3092  25 EGB.S_1
```

The format of the entries, from left to right, is:

- S indicates that the entry is a session, which includes all jobs started and completed between login and logout. J indicates a system job or a background user job that completed after the user logged out.
- Date the session or job started.
- Time the session or job started.
- Date the session or job ended.
- Time the session or job ended.
- Elapsed CPU time for the session (including all jobs started and completed within the session) or for the job. The format is *ddd/hh:mm*, *hh:mm:ss*, or *mm:ss.fff*.
- Number of disk requests made by the session or job.
- Number of jobs started and completed within the session. If the entry is a job, this number is 1.
- Username and session name.

CUSTOMIZING ACCOUNT DEFAULTS

The system provides a default login procedure, searchlist, keymap, activity, and so on. All new accounts start out using these systemwide defaults for their accounts, but users can override the defaults by creating nonstandard versions.

For information on ACL defaults for newly created users, see Chapter 4.

Default Login Procedure

The default login procedure is !Machine.Release.Current.Commands.Login'Body, and it typically looks like this:

```
with Activity;
with What;
with Common;
procedure Login is
begin
    What.Home_Library;
    Common.Create_Command;
    Activity.Set_Default
                (To_Be => "!Machine.Release.Current.Activity");
    What.Message (File  => "Daily_Message");
end Login;
```

The four procedures in !Machine.Release.Current.Commands.Login'Body cause the following to happen at login:

- The What.Home_Library procedure displays the user's home world.
- The Common.Create_Command procedure opens a command window off the window containing the home world.
- The Activity.Set_Default procedure establishes a default activity, which enables the use of subsystems.
- The What.Message procedure displays the contents of !Machine.Editor_Data-.Daily_Message (see "Creating a Daily Message" in Chapter 9 for more information).

You can customize the systemwide default login procedure (for example, to modify the file that is displayed as the daily message). To do this:

1. Edit !Machine.Release.Current.Commands.Login'Body.
2. Modify, add, or delete the appropriate procedures.
3. Promote the !Machine.Release.Current.Commands.Login'Body procedure to the coded state. If you do not promote the procedure to the coded state, the procedure cannot be executed during the login process; this will not affect users who have their own customized login procedures, but users who use !Machine-.Release.Current.Commands.Login will find that only their home world is displayed at login.

Systemwide Login Procedure

You can add a coded, parameterless procedure called System_Login to !Machine. System_Login is run before either of the other login procedures; even user ac-

counts that have been tailored so that the systemwide default login procedure is overridden by a customized login procedure will run System_Login first. For a sample application of the System_Login procedure, see the "Preparing Daily Messages in Advance" subsection in Chapter 9.

Default Searchlist

A searchlist enables the Environment to resolve command and procedure names that are executed through command windows. A systemwide default searchlist is used in any session that does not have a customized searchlist associated with it. This default searchlist is in the !Machine.Search_Lists.Default text file.

The default searchlist allows the Environment to resolve command and procedure names by searching through the following libraries:

```
$`
!Commands
!Commands.Abbreviations`
!Machine.Release.Current.Commands`
!Io
!Tools
```

The first component of the searchlist, $`, directs the Environment to look in the current library context when resolving the names of commands and procedures executed from a command window. That is, if you execute a procedure from a command window attached to a world, the procedure name is resolved by looking in that world. If you execute a procedure from a command window attached to a directory, the procedure name is resolved by looking in the world that encloses that directory.

The remaining components of the searchlist direct the Environment to look in the worlds !Commands, !Commands.Abbreviations, !Machine.Release.Current.Commands, !Io, and !Tools. That is, if the procedure name cannot be resolved by looking through the current library context, the Environment continues to search in !Commands, then !Commands.Abbreviations, and so on.

The grave symbol (`) after a world name (for example, $`, !Commands.Abbreviations`, or !Machine.Release.Current.Commands`) indicates that the links associated with the world should be searched as well as the world itself.

To customize the systemwide default searchlist (for example, to add another world where many of your installation's procedures reside), execute the Search_Lists.Edit command on the !Machine.Search_Lists.Default file.

TOKEN MANAGEMENT

Many Rational products are purchased on a per-session basis. This means that a customer purchases a specific number of concurrent usages that are authorized for a given Rational product. For example, by purchasing the Rational Environment under per-session pricing, a customer obtains a total number of concurrent logins that are authorized on the customer's R1000s. Each concurrent-usage permission is referred to as a *token*.

Once tokens have been placed on a system by Rational personnel, you can transfer them to other systems that reside within the same *site*, where the notion of a

site is defined in the Rational Price Guide: a site is essentially a set of R1000s that meet certain criteria, such as sharing a single support contract.

Caution: *If you attempt to transfer tokens between two R1000s that are not within the same site, the transfer will fail and you will lose the tokens on the donating machine. Refer to the "Restrictions" subsection for other restrictions when transferring tokens.*

For example, if your site has three R1000s and has purchased a total of 15 concurrent logins for the Environment, you can distribute the logins between those three machines (for example, Rational might authorize seven logins on one R1000 and four logins each on the others).

If an additional R1000 is purchased and placed on a separate support contract (thereby defining a different site for that R1000), none of the original 15 logins will be transferable to the new machine.

The Environment provides operations in !Commands.System_Maintenance'Spec_View.Units for managing per-session pricing. The most useful commands for a system manager are:

- Donate_Tokens, Accept_Tokens: Used for transferring tokens
- Show_Site, Show_Tokens: Used to display site and token information
- Show_Machine_Id: Used to display the system's machine identifier

Donating Tokens

The Donate_Tokens command used in conjunction with the Accept_Tokens command (which is described in the next subsection) initiates a transfer of tokens for the specified product from one R1000 to another.

When your site purchases a Rational product on a per-session basis, you obtain a specific number of tokens for that product. These tokens represent the rights of individual users to use the product; the number of tokens on a given R1000 determines the number of users who can use that product concurrently.

To execute this command, enter the Donate_Tokens command in a command window and press [Complete]. This displays:

```
Donate_Tokens (Product                => ">>PRODUCT<<",
               Donation               => 0,
               Resulting_Remote_Count => 0,
               Remote_Machine_Id      => 0,
               Remote_Site            => ">>SITE ID<<",
               Response               => "<PROFILE>");
```

The parameters specify the information required by the transfer, as follows:

- Product: Specifies the name of the product for which tokens are transferred.

  Rational supplies the appropriate name for a given product at the time of purchase. You also can list the names of the purchased products on an R1000 by executing the Show_Tokens command (see the "Displaying Token Information" subsection, later in this chapter, for a description of this command). Note that product names are case-sensitive and must be entered exactly as they are given in the Show_Tokens display.
- Donation: Specifies the number of authorized tokens to be transferred from the current machine.

- Resulting_Remote_Count: Specifies the number of authorized tokens that will exist on the accepting R1000 as a result of the transfer.
- Remote_Machine_Id: Specifies the R1000 that is to accept the transferred tokens; to obtain a value for this parameter, you can use the Show_Machine_Id command (described in a later subsection) on the accepting machine. An error results if you specify the machine-identification number of the current machine, because the current machine cannot donate tokens to itself.
- Remote_Site: Specifies the site to which the accepting machine belongs. To obtain a value for this parameter, you can use the Show_Site command (described in a later subsection) on the accepting machine.

  The specified site must match the site of the donating machine; you may want to run the Show_Site command on both machines to verify that both the donating and the accepting machine have the same site-identification number.
- Response: Specifies the system response in case of errors. The default specifies the job response profile.

Executing the Donate_Tokens command results in a display of an Accept_Tokens procedure with the appropriate parameter values filled in. The transfer operation is completed when you copy the Accept_Tokens call to the accepting R1000 and execute it there (see the "Accepting Tokens" subsection, below, for a more detailed description of the Accept_Tokens command).

The Donate_Tokens procedure records the initiated transfer in the error log of the donating R1000.

Accepting Tokens

The Accept_Tokens command completes the transfer of tokens for the specified product from one R1000 to another. Executing the Donate_Tokens command displays the Accept_Tokens command with specific parameter values automatically filled in. The resulting parameters are shown below:

```
Accept_Tokens (Product         => ">>PRODUCT NAME<<",
               Donation        => 0,
               Resulting_Count => 0,
               Code            => ">>AUTHORIZATION CODE<<",
               Authorization   => "",
               Response        => "<PROFILE>");
```

The parameters specify the information required to complete the transfer, as follows:

- Product: Specifies the name of the product for which tokens are being accepted. If the product has not been authorized previously, accepting tokens will authorize it. The exact name that can be specified for a given product is supplied by Rational when that product is purchased. You also can list the names of the purchased products on a given R1000 by executing the Show_Tokens procedure. It is important to note that product names are case-sensitive and must be entered exactly as they are given in the Show_Tokens display.
- Donation and Resulting_Count: Specify values that reflect the input that was given to the Donate_Tokens command. These parameter values specify the number of transferred tokens and the total number of tokens on the accepting machine, respectively.
- Code and Authorization: Code specifies a value that is a unique authorization code valid only for the current day, on the current machine, at a particular site.

This code is generated by executing the Donate_Tokens command. When the Donate_Tokens command is used, the resulting Accept_Tokens command will have an empty Authorization parameter, and the Accept_Tokens command must be run on the same day as Donate_Tokens. The Authorization parameter is used in place of the Code parameter when tokens are purchased or transferred from the home office. This functions the same way as the Code parameter but is not date-dependent. It can be used at any time, but only once. The system keeps track of Authorization parameters used, so every one generated by Rational for the same machine must be different.

- Response: Specifies the system response in case of errors. The default specifies the job response profile.

You must copy and execute the Accept_Tokens command on the R1000 that is accepting the token(s).

As a result, the new tokens are made available immediately, and the new number of authorized tokens is recorded in the accepting machine's error log. See the "Sample Token-Transfer Operation" subsection, below, for a sample token transfer.

Restrictions

The transfer of tokens is subject to stringently enforced restrictions, as described below:

- The Accept_Tokens command must be executed on the same calendar day (12 A.M. to 12 P.M.) as the Donate_Tokens command that generated it. *If this condition is not met, the donated tokens will be lost and can be recovered only by contacting the Rational Response Center.*
- The donating and accepting machines must reside within a single site. The Donate_Tokens command will fail if the specified remote site is different from the site of the current machine. *If this condition is not met, the donated tokens will be lost and can be recovered only by contacting the Rational Response Center.*
- The Donate_Tokens command fails if the transfer would leave the donating machine with a negative number of tokens.
- The donating machine cannot accept tokens on the same calendar day (even if the machine is accepting tokens for a different product than was transferred by the donating machine's tokens).
- The accepting machine cannot donate tokens on the same calendar day.
- Machines that are not set up for per-session pricing cannot donate or accept tokens.
- Execution of the Donate_Tokens and Accept_Tokens commands requires operator capability. (Refer to Chapter 4 for a description of operator capability.)
- The sum of the donated tokens and the tokens already on the receiving machine must exactly match the value of the Resulting_Count parameter of the Accept_Tokens command.

Sample Token-Transfer Operation

Assume that you have two R1000s at your site and a total of ten authorized logins that are currently distributed so that each R1000 has five. Assume further that you

want to transfer three logins from one machine to the other, resulting in eight logins on the other machine. Use the Show_Site command to obtain the site ID of the remote machine (you might also want to use the Show_Site command on the current machine to verify that the site IDs of the donating and accepting machines are identical).

The Donate_Tokens command initiates the transfer:

```
Donate_Tokens (Product                => "Login",
               Donation               => 3,
               Resulting_Remote_Count => 8,
               Remote_Machine_Id      => Destination_Machine_Id,
               Remote_Site            => "Site_Id");
```

After executing this command, the Accept_Tokens command is displayed in the current output window. This procedure includes the authorization code, which is needed to complete the transfer:

```
Accept_Tokens (Product         => "Login",
               Donation        => 3,
               Resulting_Count => 8,
               Code            => "A_Unique_Authorization_Code");
```

To complete the transfer, you now need to copy the displayed Accept_Tokens procedure into a file, archive it to the destination machine, and enter it in a command window there.

Displaying Token Information

This procedure shows token information for the specified user or product. You can use this procedure after a token transfer to verify that the current machine has the correct token limit. You can also use this command to compare actual usage with purchased usage. When no more tokens are available for a given product, end users who want to use that product can use this command to find out which sessions have the tokens. This procedure has the following parameters:

```
Show_Tokens (User_Filter        => "",
             Product_Filter     => "",
             Include_Sessions   => True,
             Include_Debugging  => False,
             Include_Statistics => False);
```

By default, this command displays the following fields of information for every token-managed product:

- Users: Lists the user sessions that currently have tokens.
- Limit: Shows the number of authorized tokens that have been placed on the current machine.
- Current: Shows the number of tokens that are currently in use (one for each of the sessions listed in the Users field).
- Buy_Out: A *buy-out* number exists for each product. The buy-out number is the maximum number of tokens that can be placed on a single system, after which token enforcement is no longer performed. Rational establishes this number on a product-by-product basis.

In the following example, Environment logins have a buy-out number of 12; this means that if a customer places 12 login tokens on the same system, that system

will allow unlimited concurrent logins. Because the Limit and Buy_Out fields are equal for Login, there is no limit on the number of concurrent logins. However, for the remaining products, the maximum usage is set by the Limit field.

You can use the User_Filter and Product_Filter parameters to specify a particular username or a particular product name for which to display token information. You can set the Include_Sessions parameter to False to cause the display to omit the information in the Users field. The remaining parameters (Include_Debugging and Include_Statistics) are for Rational use only.

An example of token information displayed by the Show_Tokens command is shown below:

```
Product   Users                Limit Current Buy_Out
------- ------------------ -----------------
Login                          12     4    12
          SJL.S_1
          RJS.S_1
          JIM.S_1
          PHIL.S_1
X Interface                    10     2    12
          SJL.S_1
          RJS.S_1
Design_Facility                 8     1    12
          RJS.S_1
```

Displaying the Machine Identifier

Enter `Show_Machine_Id` in a command window to display the unique machine-identification number of the current machine. An error results if you specify the machine-identification number of the current machine to the Donate_Tokens procedure (the current machine cannot donate tokens to itself).

Displaying the Site Identifier

The Show_Site command displays the site-identification number of the current machine. To donate tokens from one machine to another, you should verify that the donating and accepting machines both have the same site identification. To execute this procedure, enter `Show_Site` in a command window. If your machine has not been assigned a site identification, executing this command displays:

```
Site has not been set.
```

4 Access Control

Access control protects data against unauthorized access. Controlling access to a world, file, or Ada unit means controlling who can read that object, write to that object, and/or delete that object. System managers, project leaders, and individual users control who has the right to see, change, create, and delete objects in their various areas of responsibility. This chapter describes the process for granting and removing various *access rights* to specific *groups* of users of the worlds, files, and Ada units in the Environment.

Before a system user can access an object created by someone else, access rights must be specifically granted to the group of which that system user is a member. Access rights are specific. For example, giving a group the Read access right (or simply *read access*) to an object allows any member of the specified group to view the object and, if the object is an Ada unit, to execute the object. However, granting read access to a group does not grant that group the right to modify the object. (*Write access* is required to modify an object.) A group does not have a specific kind of access right to an object unless that particular access right is explicitly granted. Thus, access control is a means of preventing unauthorized users from viewing or changing specific Environment objects.

Access control regulates not only who can see and change objects but also who can perform certain system operations that access those objects. The system operations affected by access control include:

- Name resolution
- Links, compilation, and execution
- Networking
- CMVC and subsystems (see the "CMVC and Subsystems" subsection at the end of this chapter and the note below)
- Archive commands

For more details about the impact of access control on each of these types of operations, see the "Implications of Access Control" section at the end of this chapter.

For a more detailed description of the procedures used for access control, see the "Procedures for Access Control" section later in this chapter. For full descriptions of most access-control commands, see the description of package !Commands-.Access_List in the Library Management (LM) book of the *Rational Environment Reference Manual.*

Note: *When using CMVC and subsystems, see the Project Management (PM) book of the* Rational Environment Reference Manual *for information about package Cmvc_Access_Control.*

ACCESS-CONTROL LISTS

Access-control settings are recorded in *access-control lists* (ACLs). Every world, file, and Ada unit has an ACL associated with it. An object's ACL contains *entries* in the form: *group_name=>access_rights*. Each entry grants one or more access rights to the members of a given group. For example, the entry "Engineering=>RW" grants read and write access to members of the group named Engineering. The entry "Tech_Support=>R" grants read access and denies write access to members of the group named Tech_Support.

An object's ACL can contain up to seven entries and up to 512 characters. Multiple entries are separated by commas. For example, the ACL "Engineering=>RW, Tech_Support=>R" has two entries. One entry grants read and write access to members of the group Engineering. The other entry grants read-only access to members of the group Tech_Support.

Only three types of objects have ACLs:

- Worlds
- Files
- Ada units

Access to any directory is controlled by the ACL of the world enclosing that directory. For a full description of the effect of access rights on different types of objects, see the next section in this chapter, "Access Rights."

ACLs for New Objects

New objects are given their ACLs according to the following rules:

- A new world is given the ACL of the enclosing world.
- A new file or Ada unit is given the *default* ACL of the enclosing world.
- A new version of a file or Ada unit is given the ACL of the previous version.
- If you have *owner* access rights to the enclosing world (owner access is defined in the next section, "Access Rights"), you can modify the ACL of any new object in that world.

Viewing and Modifying ACLs

If you have owner access rights to an ACL, you can run the following commands that act on the ACL:

- Set: Set or remove the access rights of groups
- Add: Grant additional access rights to groups
- Display: Display the current access rights of groups
- Set_Default: Specify the default ACL for a world
- Add_Default: Add one or more entries to the default ACL
- Default_Display: Display a world's default ACL

For a more detailed description of the first three commands and how to use them, see the "Procedures for Access Control" section. For a more detailed description of

the last three commands and how to use them, see the "Management of Default Access Lists" section.

For a description of systemwide ACLs, see the "Systemwide Access Control" section, later in the chapter.

Note that certain kinds of locks provide additional system security for files and Ada units. Only the user who holds the lock can change the ACLs of a locked file or Ada unit. For details about locks, see the next subsection, "Editor Locks, Versions, and ACLs."

Editor Locks, Versions, and ACLs

Commands (such as Set and Add) that change ACLs are sensitive to read locks and write locks. In particular, certain kinds of locks prevent ACLs from being changed on files or Ada units, and other kinds of locks allow ACLs to be changed but cause new versions to be created in the process. As a general rule:

- You can change the ACL of a text or Ada object when you or other users hold a read lock on that object.
- You can change the ACL of a text or Ada object when you hold the write lock on that object.
- You cannot change the ACL of a text or Ada object when another user holds the write lock on that object. (Use the What.Locks command to display the locks that exist on a specified object. See package What in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual* for more information about the What.Locks command.)

Changing the ACL of a locked Ada unit has some side effects. Specifically, if you change the ACL of an Ada unit on which you hold a read or a write lock, a new version of that unit is created on which the new ACL is set. Furthermore, your read or write lock is automatically transferred to the new version; the previous version with the old ACL becomes a deleted version. Similarly, if you change the ACL of an Ada unit on which another user holds a read lock, a new version is created on which the new ACL is set. However, the other user continues to view the previous version, which automatically becomes a deleted version.

The existence of deleted versions occasionally causes commands to fail unexpectedly because of the out-of-date ACLs on those versions. If you have access rights to a deleted version, and own any locks on it, you can prevent these problems by destroying that deleted version by running the Library.Expunge command.

ACCESS RIGHTS

Access rights for files and Ada units differ from access rights for worlds. These differences are described below.

Worlds

Worlds can have one or more of the following access rights:

- Read (R): Permits the contents of a world to be viewed; permits names within a world to be resolved

- Create (C): Permits new objects to be created in a world; permits new versions of existing objects to be created in a world
- Delete (D): Permits a world to be deleted
- Owner (O): Permits a world's links, library-switch associations, ACL, and default ACL to be changed; also permits the ACLs of objects in the world to be changed and permits objects in the world to be frozen or unfrozen

To view a world, you must have read access to that world, to every world that encloses it, and to the switch file associated with that world.

If you attempt to view a world for which you do not have read access or if you attempt an operation on an object for which you do not have the appropriate create, delete, or owner access, the system displays an information message on your terminal. Furthermore, if you try to reference an object in a world for which you do not have read access, the Environment acts as if the object does not exist.

Files and Ada Units

Files and Ada units can have one or both of the following access rights:

- Read (R): Permits the current contents of an object to be inspected; permits Ada units to be executed
- Write (W): Permits an object to be opened for editing or deleted; permits Ada units to be promoted or demoted

To view a file or an Ada unit, you must have read access to it and also have read access to every world that encloses it. Furthermore, to edit a file or an Ada unit, you must have write access to it, as well as create access to the world enclosing it (see "Worlds," above). To open an object for editing, you must have write access to the object. To create new versions of an object, you must have create access to the enclosing world. All these access rights are automatically set by the Environment at the time a file or Ada unit is created. However, these access rights can be changed at any time by anyone with the proper access rights.

If you attempt to view or execute an object to which you do not have read access or if you attempt to modify an object to which you do not have write access, the system displays an information message on your terminal.

Directories

Directories do not have their own ACLs. Requests to view or change directories are granted or denied according to the ACL of the enclosing world.

- Creating a new object in a directory requires create access to the directory's enclosing world.
- Changing the ACL for an object in a directory requires owner access to the enclosing world.

Mixing Access Rights

Mixed access rights refers to an ACL entry that includes access rights both to worlds and to files and Ada units. For your convenience, when you specify an ACL

for a file or Ada unit, the Set and Add commands ignore any access rights that are not applicable to files and Ada units. For example, the following ACL contains a mixture of access rights: Public=>RWCOD. If this ACL is specified for a file or Ada unit, the Set and Add commands ignore the world-related rights C, O, and D. Likewise, if this ACL is specified for a world, the Set and Add commands ignore the nonworld access right W. This allows you to make a single entry in an ACL to set access rights for multiple objects of various kinds.

As a further convenience, the access rights Delete and Write are interchangeable. Thus, if write access (W) is specified for a world, it grants delete access (D) to files and Ada units. Likewise, if delete access (D) is specified for a file or Ada unit, it grants write access (W) to a world.

Overriding Access Rights

Members of a special predefined group called *Privileged* can access any object regardless of its ACL. To activate this capability, a member of this group must execute the !Commands.Operator.Enable_Privileges command. For details, see the "Privileged" subsection, below. In the standard Environment, username *Operator* is a member of the Privileged group.

Note that privileged mode (described in the "Privileged" subsection) is not the same thing as operator capability (described in the "Operator" subsection.)

DESCRIPTIONS OF GROUPS

A group is a set of one or more usernames to which access rights can be granted. Every user automatically belongs to at least one group, a group that has the same name as the user. For example, the username *John* defines a group called John. This allows access rights to be granted to individual users in a straightforward way, as in the example ACL entry: John=>R.

In addition, the usernames for all users are added by default to two special, predefined groups—namely, Public and Network_Public. If you grant access rights for an object to either of these two groups, you are giving all users access to that object. See "Public" and "Network_Public" in the "Special Groups" subsection, below. Note that although any username can be removed from the Public and/or Network_Public groups, such removal is not recommended, because it locks the removed user out of essential worlds.

Groups cannot reference other groups, although there is no limit to the number of usernames that can belong to a group. Furthermore, there is no limit to the number of groups to which a username can belong. For example, user John may be a member of groups John, Public, Network_Public, and Engineering. Up to 1,000 user-defined groups can exist on a single R1000 at a time.

When a *job* (a command or program) is promoted, it usually has to read other objects. It is granted read access to a required object if the user initiating the job is a member of at least one group that has read access to that object. This is true because every job has an *identity* that is the username of the user who initiated the job. (Note that when a job is initiated by the !Commands.Program.Create_Job or !Commands.Program.Run_Job command, the identity can be set in the Options parameter. Thus, a user who lacks the required read access to required objects can run a job by specifying in the Options parameter a user who does have the re-

quired read access. Also, a job can change its own identity by calling the !Commands.Program.Change_Identity command.)

Special Groups

There are four special groups. They are listed and briefly described in Table 4-1. Each of these four groups is described in greater detail in separate subsections on the following pages.

Table 4-1 Special Groups

Group	Description
Operator	Allows users to perform operations within the Environment that require *operator capability*
Privileged	Allows users to perform operations without the restrictions of access control
Public	Gives everyone on a system access to a world
Network_Public	Gives usernames access to objects on other machines (on the same network)

Operator

Certain Environment commands can be executed only by users that have *operator capability*. All members of the group Operator have operator capability. Furthermore, additional users can be given operator capability without being added to the Operator group. This allows you to grant permission to specified nonoperator users to run certain restricted Environment commands without granting them access to the worlds, files, and Ada units to which the Operator group has access.

To give a user operator capability without adding this user to the Operator group, you must grant this user write access (W) to the !Machine.Operator_Capability file. (For a description of write-access control, see the "Access Rights" section, above.)

Operator capability should not be confused with *privileged mode*. Whereas privileged mode enables users to override access control, giving them access to all objects, operator capability enables users to execute any of a specific, restricted set of commands. Note, however, that users who can enable privileged mode can also:

- Execute commands requiring operator capability, since privileged mode temporarily grants access to the !Machine.Operator_Capability file
- Give themselves or others permanent operator capability by granting themselves access to the !Machine.Operator_Capability file through the ACL
- Add themselves or others to the !Machine.Groups.Operator group

(See "Privileged," below. Also see package Operator in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* for more details about operator capability.)

Finally, note that many of the commands described in package Operator can be run from the operator's console command interpreter. (For a detailed description of these commands, see Appendix A.)

Privileged

The Environment provides another predefined group called *Privileged*, whose members can execute jobs in *privileged mode*. Privileged mode overrides access control; thus, jobs executed in privileged mode have complete access to all Environment objects. To execute a job in privileged mode, a member of the Privileged group must execute the !Commands.Operator.Enable_Privileges command as part of the job (that is, in the same command window). After the job has finished executing, privileged mode is automatically disabled.

Note that members of the Privileged group can execute any Environment command in privileged mode, including commands that require operator capability, because privileged mode grants access to the !Machine.Operator_Capability file (see "Operator," above).

Public

The Public group includes all users on a system. When a new username is created, the user automatically becomes a member of the user's own group, the Public group, and the Network_Public group. Because all users are members of the Public group, giving this group access to a world, file, or Ada unit gives every user on the system access to that world, file, or Ada unit. At sites where users have nonrestricted access rights, all worlds, files, and Ada units can include group Public in their ACLs, effectively giving everyone access to everything on a system.

Network_Public

The Network_Public group is used at sites that use Rational Networking—TCP/IP. Adding the Network_Public group to ACLs gives all usernames access to objects on other machines on the same network.

Although usernames are added to these groups by default, they can be explicitly removed from these groups by using the Remove_From_Group command (see the "Removing a User from a Group" subsection, below).

User-Defined Groups

Package Operator allows users with operator capability to create groups for access control. Using the Operator.Create_Group command (described in the "Procedures Related to Groups" section, below), a user with operator capability can define a group that grants access to a shared set of objects for a common set of users. For instance, the operator could create a user-defined group called Engineering. After group Engineering has been created, usernames Bill and Ann can be added to the group. Each username has its own group consisting of its username and each is also a member of group Engineering for any access they want to share.

For a detailed description of the operations in package Operator, see package Operator in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*. Also see "Operator" under the "Special Groups" subsection, above.

PROCEDURES RELATED TO GROUPS

The following subsections describe how to:

- Create a group
- Delete a group
- Add a user to a group
- Remove a user from a group
- Display the members of a group

These group-related procedures are an important component of access control. One way to grant access rights to a particular user is to add that user to a predefined group that has already been given access rights to the objects in question.

For instance, if you wanted to grant a user operator capability, one way to do this would be to add this user to the Operator group by entering the Add_To_Group procedure in a command window, completing it, and then promoting it. Each of the group-related commands is described in a separate subsection below.

Creating a Group

To create a group, use the !Commands.Operator.Create_Group procedure. The specification for this procedure is:

```
procedure Create_Group (Group    : String := ">>GROUP NAME<<";
                        Response : String := "<PROFILE>");
```

The Group parameter specifies the name of the group to be created.

The Response parameter specifies the system response in case of errors in executing the command. The default specifies the session job response profile. For more information about the Response parameter, see the "Parameter-Value Conventions" section in the Reference Summary (RS) book of the *Rational Environment Reference Manual*.

You can run this command in a command window or from the operator's console command interpreter. For example, to create a group called Engineering, execute the following command:

```
Operator.Create_Group ("engineering")
```

Deleting a Group

To delete a group, use the !Commands.Operator.Delete_Group procedure. The specification for this procedure is:

```
procedure Delete_Group (Group    : String := ">>GROUP NAME<<";
                        Response : String := "<PROFILE>");
```

The Group parameter specifies the name of the group to be deleted.

The Response parameter specifies the system response in case of errors in executing the command. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to delete a group called Engineering, enter the following command:

```
Operator.Delete_Group ("engineering")
```

Adding a User to a Group

To add a user to a group, use the !Commands.Operator.Add_To_Group procedure. The specification for this procedure is:

```
procedure Add_To_Group (User     : String := ">>USER NAME<<";
                        Group    : String := ">>GROUP NAME<<";
                        Response : String := "<PROFILE>");
```

The User parameter specifies the name of the user to be added to the group.

The Group parameter specifies the name of the group to which to add the user.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to add a user named Bill to a group called Engineering, execute the following command:

```
Operator.Add_To_Group ("bill","engineering")
```

Removing a User from a Group

To remove a user from a group, use the !Commands.Operator.Remove_From-_Group procedure. The specification for this procedure is:

```
procedure Remove_From_Group
             (User     : String := ">>USER NAME<<";
              Group    : String := ">>GROUP NAME<<";
              Response : String := "<PROFILE>");
```

The User parameter specifies the name of the user to be removed from the group.

The Group parameter specifies the name of the group from which the user will be removed.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to delete a user named Bill from a group called Engineering, execute the following command:

```
Operator.Remove_From_Group ("bill","engineering")
```

Displaying the Members of a Group

To display the members of a group, use the !Commands.Operator.Display_Group procedure. The specification for this procedure is:

```
procedure Display_Group (Group    : String := ">>GROUP NAME<<";
                         Response : String := "<PROFILE>");
```

The Group parameter specifies the name of the group to be displayed.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to display the members of a group called Engineering, execute the following command:

```
Operator.Display_Group ("engineering")
```

PROCEDURES FOR ACCESS CONTROL

Procedures in package Access_List can be grouped according to their functional purpose, as listed in Table 4-2.

Table 4-2 Procedures in Package Access_List

Functional Purpose	Procedures
Access rights for files and Ada units	Read, Write
Access rights for worlds	Create, Delete, Owner, Read
Setting or removing access rights	Set, Set_Default
Granting additional access rights	Add, Add_Default
Displaying access rights	Display, Default_Display

Use the procedures described in this section ("Procedures for Access Control") to manage ACLs for files and Ada units. Use the procedures described in the next section ("Management of Default Access Lists") to manage default ACLs for worlds. For an overview of access rights, see the "Access Rights" section, above.

Setting the ACL for an Object

When objects are created, they are assigned ACLs according to the rules described in the "ACLs for New Objects" subsection. If an object does not have the desired ACL, you can change it using the !Commands.Access_List.Set procedure. The specification for this procedure is:

```
procedure Set (To_List    : Acl    := "Network_Public => RWCOD";
               For_Object : Name   := "<SELECTION>";
               Response   : String := "<PROFILE>");
```

The To_List parameter specifies the new ACL that you want the object to have—for example, "John=>RW, Mary=>R".

The For_Object parameter specifies the object whose ACL you want to set.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to set the ACL for an object called !Users-.John.File_1, execute the following command:

```
Access_List.Set
    ("john=>rw,bill=>r,engineering=>r","!users.john.file_1")
```

Adding an Entry to an Object's ACL

If you need to update the ACL for an object with a new entry, you can use the !Commands.Access_List.Add procedure rather than resetting the entire ACL. The specification for this procedure is:

```
procedure Add (To_List    : Acl    := "Network_Public => RWCOD";
               For_Object : Name   := "<SELECTION>";
               Response   : String := "<PROFILE>");
```

The To_List parameter specifies the entry that you want to add to the object's current ACL—for example, "John=>RCOD,Mary=>R".

The For_Object parameter specifies the object to whose ACL you want to add an entry.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to add John=>RCOD to the current ACL for a world called !Users.Engineering, execute the following command:

```
Access_List.Add ("john=>rcod","!users.engineering")
```

Displaying the ACL for an Object

To display the ACL for an object, you can use the !Commands.Access_List.Display procedure. The specification for this procedure is:

```
procedure Display (For_Object : Name := "<CURSOR>");
```

The For_Object parameter specifies the object whose ACL you want to display.

You can run this command in a command window or from the operator's console command interpreter. For example, to see the ACL for an object called !Users-.John.File_1, execute the following command:

```
Access_List.Display ("!users.john.file_1")
```

MANAGEMENT OF DEFAULT ACCESS LISTS

As described in the "ACLs for New Objects" subsection, above, every world has a *default ACL* associated with it. This default ACL is given to new files and Ada units created within the world. If you have owner access to the enclosing world, you can modify the ACLs that were inherited by the new objects, in addition to any other ACLs in the world. The procedures below allow you to set, add an entry to, and display the default access list of a world.

Setting the Default ACL for a World

When new objects are created within a world, they are assigned the default ACL for that world. If a world does not have the desired default ACL, you may want to change it using the !Commands.Access_List.Set_Default procedure. The specification for this procedure is:

```
procedure Set_Default
              (To_List   : Acl    := "Network_Public => RW";
               For_World : Name   := "<SELECTION>";
               Response  : String := "<PROFILE>");
```

The To_List parameter specifies the new ACL that you want the object to have—for example, "John=>RW,Mary=>R".

The For_World parameter specifies the world whose default ACL you want to set.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to set the default ACL to give John read and write access and Bill read access for a world called !Users.John.World_1, execute the following command:

```
Access_List.Set_Default("john=>rw,bill=>r,
                        "!users.john.world_1")
```

Adding an Entry to a World's Default ACL

If you need to update the default ACL for a world with a new entry, you can use the !Commands.Access_List.Add_Default procedure rather than resetting the entire ACL. The specification for this procedure is:

```
procedure Add_Default
              (To_List   : Acl    := "Network_Public => RW";
               For_World : Name   := "<SELECTION>";
               Response  : String := "<PROFILE>");
```

The To_List parameter specifies the entry that you want to add to the object's current ACL—for example, "John=>RW, Mary=>R".

The For_World parameter specifies the world to whose default ACL you want to add an entry.

The Response parameter specifies the system response in case of errors. The default specifies the session job response profile.

You can run this command in a command window or from the operator's console command interpreter. For example, to add an entry of John=>RW to the current ACL of Engineering=>RW to the default ACL for a world called !Users.Engineering, execute the following command:

```
Access_List.Add_Default ("john=>rw","!users.engineering")
```

Displaying the Default ACL for a World

To display the default ACL for new objects created within a world, you can use the !Commands.Access_List.Display_Default procedure. The specification for this procedure is:

```
procedure Default_Display (For_World : Name := "<CURSOR>");
```

The For_World parameter specifies the world whose default ACL you want to display.

You can run this command in a command window or from the operator's console command interpreter. For example, to see the default ACL for a world called !Users.John.World_1, execute the following command:

```
Access_List.Default_Display ("!users.john.world_1")
```

SYSTEMWIDE ACCESS CONTROL

The standard Environment release is configured with a default set of access rights. These access rights control the access to system files located primarily in !Machine. The degree of access allowed to the standard Environment can be controlled using the Set_Universe_Acls command. This command allows you to restrict access to worlds, files, and Ada units, usually for security reasons.

The access rights of the standard Environment can be monitored using the Check_Universe_Acls command. Problems with the ACLs of the standard Environment are displayed as errors. This command can be used to diagnose some problems that affect system operation (for example, problems in sending mail).

If errors are displayed when you execute the Check_Universe_Acls command, you can correct these problems using the procedures described earlier in the "Procedures for Access Control" section. If a large number of errors are displayed, you might be able to resolve some of these systemwide problems using the Set_Universe_Acls command. However, if system operation is affected by severe access-control problems, you may not be able to execute the Set_Universe_Acls command in a command window. Refer to the "Limitations of Set- and Check_Universe_Acls" subsection, below, for more information.

The Set_Universe_Acls and Check_Universe_Acls commands are described in greater detail below.

Setting Standard Environment ACLs

The access rights of the standard Environment are set at the time of system shipment. You can use the Set_Universe_Acls command to tailor the access rights of files, Ada units, and worlds to the level of security appropriate for your site.

Note: *Depending on the level of security you choose, this command might prevent users from accessing a large number of worlds, Ada units, or files. Therefore, it is recommended that you consult with your Rational technical representative before using this command.*

To run this procedure, type `Set_Universe_Acls` into a command window and press [Complete]. Something similar to the following is displayed:

```
Set_Universe_Acls (Level                => 0,
                   Implementation_Okay  => True,
                   Network_Read_Okay    => True,
                   Network_Write_Okay   => True,
                   Trace_Only           => False,
                   Produce_Tables       => False,
                   Tables_Output_File   => "acl_tables");
```

The parameters for this procedure are described below:

- Level

  The default level is 0, but this parameter accepts numbers in the range of 0–3. The meanings of the level numbers are:

  – Level 0 (None)

    This level places no access-rights restrictions on users. For instance, all users are able to edit system configuration files, machine initialization files, name maps, operator commands, and other files and Ada units that could disrupt system operation. Table 4-3 shows the access rights of worlds in the standard Environment when the security level is set to level 0 (no restrictions).

Table 4-3 World ACLs, Level 0 (None)

Pathname	Public	Network_Public	Operator	System
!		RCO		
![Commands,Io,Lrm,Model.??]		RCO		
!Tools		RCO		
!Compiler_Interface		RCO		
!Commands.Abbreviations		RCO		
!Implementation		RCO		
!Users		RCO		
!Machine		RCO		
!Machine.[Cg_Data,Devices,Editor_Data]		RCO		
!Machine.Accounting		RCO		
!Machine.Error_Logs		RCO		
!Machine.[Groups,Users]		RCO		
!Machine.[Mail,Transfer]??		RCO		
!Machine.Release??		RCO		
!Machine.Search_Lists		RCO		
!Machine.Temporary		RCO		
!Machine.[Sims,Sims.Master,Sims.To_Rational]		RCO		
!Machine.Sims.[Database,Detail]??		RCO		
!Machine.Sims.To_Rational??		RCO		

- Level 1 (Open)

  This level also allows all users to gain nonrestricted access; objects are protected, but any user can change ACLs to gain access to all files, Ada units, and worlds.

- Level 2 (Safe)

  This level is designed to protect users and the system. Only a user logged in as Operator can change ACLs in order to create new, nonuser libraries or to gain access to the worlds, Ada units, or files that are accessible to all users when the security levels are set at level 1 or level 2.

- Level 3 (Secure)

  This level is like level 2, except that there is less network access and read access. Level 3 is about as restrictive as the system can be and still run. It prevents most users, other than members of the Operator group, from successfully executing operator commands—even if users have given themselves operator capability by granting themselves write access to the !Machine-.Operator_Capability file. (Refer to the "Operator" subsection earlier in this chapter for a description of operator capability.)

  Table 4-3 shows that all users have been given read, create, and owner access to all worlds. This occurs because members of all groups (including members of the Operator, System, and Public groups) are also members of the Network_Public group.

  This nonrestricted level of access control contrasts with the access rights of the same worlds when the security level is set to level 3 (secure), as in the example shown in Table 4-4. Table 4-4 shows that when the level parameter is set to 3, only members of the Operator group have read, create, and owner access to all the worlds. All other users have no access or restricted access to these same worlds.

  Table 4-4 shows the access rights of worlds in the standard Environment when the security level is set to level 3.

  Set the Level parameter to whatever degree of access-rights control you decide is appropriate for your site. Also, if users encounter problems accessing the system (for instance, if they cannot log in or send messages), you might be able to resolve the problems by running the Set_Universe_Acls command with its Level parameter set to any of the levels described above. (Refer to the "Limitations of Set- and Check_Universe_Acls" subsection, below, for more information.)

- Implementation_Okay

  By default, this parameter is set to True so that access is given to !Implementation and to !Compiler_Interface. (Note that !Compiler_Interface is readable because it contains the switch file for the standard Environment.)

- Network_Read_Okay

  By default, this parameter is set to True so that the Network_Public group is granted read access to most files, Ada units, and worlds (except when level 3 is specified).

- Network_Write_Okay

  By default, this parameter is set to True. It is similar to the Network_Read_Okay parameter, except that it affects write access.

Table 4-4 World ACLs, Level 3 (Secure)

Pathname	Public	Network_Public	Operator	System
!	R	R	RCO	
![Commands,Io,Lrm,Model??]	R		RCO	RC
!Tools	R	R	RCO	RC
!Compiler_Interface		R	RCO	
!Commands.Abbreviations	R		RCO	RC
!Implementation	R		RCO	
!Users	R		RCO	
!Machine		RC	RCO	
!Machine.[Cg_Data,Devices,Editor_Data]		R	RCO	RC
!Machine.Accounting			RCO	RC
!Machine.Error_Logs		RC	RCO	RC
!Machine.[Groups,Users]		R	RCO	RC
!Machine.[Mail,Transfer]??			RCO	
!Machine.Release??		R	RCO	
!Machine.Search_Lists	RC		RCO	R
!Machine.Temporary		RC	RCO	
!Machine.[Sims,Sims.Master,Sims.To_Rational]	RC		RCO	
!Machine.Sims.[Database,Detail]??	RC		RCO	
!Machine.Sims.To_Rational??	RC		RCO	

Note: *When you add a new user account to the system, you may need to update the !Machine.User_Acl_Suffix file and the !Machine.User_Default_Acl_Suffix file, so that the new user will have the access rights you want.*

- Trace_Only

  By default, this parameter is set to False. Set this parameter to True if you want to execute a trial run of the command only (refer to the "Sample Displays from Set_Universe_Acls" subsection, below, for a more detailed description of this option).

- Produce_Tables

  By default, this parameter is set to False. When it is set to True, all other parameters are ignored, and all combinations of ACLs are written to the file Tables_Output_File. (These tables are for Rational documentation purposes.)

- Tables_Output_File

  This parameter writes the contents of the Tables_Output_File file into the named file. By default, the named file is Acl_Tables.

Sample Displays from the Set_Universe_Acls Command

Executing the Set_Universe_Acls command with a specified security level and parameter Trace_Only set to True displays the access rights of all worlds in the standard Environment for the specified level.

For example, suppose you decide to set the ACLs at your site to level 3 so that security is at a maximum. The sample display below shows the result of executing the Set_Universe_Acls command at this level of security with the Trace_Only parameter set to True:

```
------------------------------------------------------------------------------------------
!USERS.OPERATOR % SET_UNIVERSE_ACLS                                    STARTED 2:58:42 PM
------------------------------------------------------------------------------------------

!'c(world)                          : Public=>R,Operator=>RCO,Network_Public=>RC
![Commands,Io,Lrm,Model??]'c(world): Public=>R,System=>RC,Operator=>RCO,Network_Public=>R
!tools'c(world)                    : Public=>R,System=>RC,Operator=>RCO,Network_Public=>R
!compiler_interface'c(world)       : Operator=>RCO,Network_Public=>R
!commands.abbreviations'c(world)   : Public=>R,System=>RC,Operator=>RCO,Network_Public=>R
!Implementation'c(world)           : Public=>R,Operator=>RCO,Network_Public=>R
!users'c(world)                    : Public=>R,Operator=>RCO,Network_Public=>RC
!machine'c(world)                  : Operator=>RCO,Network_Public=>RC
   ...

!Machine.Accounting.@'c(~library) : System=>RW,Operator=>RW
!Machine.Error_Logs.[@,~Mail_Distribute_Server@]'c(~library) : System=>RW,Operator=>RW
!Machine.Error_Logs.Mail_Distribute_Server@'c(~library)
                                    : System=>RW,Operator=>RW,Network_Public=>RW
!Machine.Machine_Name'c(~library) : Public=>R,System=>R,Operator=>RW,Network_Public=>R
!Machine.[Mail,Transfer].[??,~Distribute.??]'c(~library)     : Operator=>RW
!Machine.[Mail,Transfer].Distribute.??'c(~library)            :
                                                  Operator=>RW,Network_Public=>RW

!Machine.release.??'c(file,ada)'c(~library)                   :
                                                  Operator=>RW,Network_Public=>R
!Machine.Search_lists.@'c(~library)                           : Public=>RW,Operator=>RW
!Machine.sims.@'c(ada,file)'c(~library)                       : Public=>RW,Operator=>RW
!Machine.sims.[master,to_rational].@'c(ada,file)'c(~library)  : Public=>RW,Operator=>RW
!Machine.sims.[database,detail]@'c(ada,file)'c(~library)      : Public=>RW,Operator=>RW
!Machine.sims.to_rational.@'c(ada,file)'c(~library)           : Public=>RW,Operator=>RW
!machine.temporary.@'c(~library)                              : Network_Public=>RW
Add !Machine.[Mail,Transfer].??'c(~library)                   : Mailer=>RW
Add !Users.@.Mailbox.Main_@'c(~library)                       : Mailer=>RW}
```

This small portion of a sample output shows the access rights for all worlds in the standard Environment. After you determine that the ACLs are appropriate for your desired level of security, you can execute the Set_Universe_Acls command again, but this time with the Trace_Only parameter set to False. The sample display below shows the access-rights settings when level 3 is specified:

```
------------------------------------------------------------------------------------------
!USERS.OPERATOR % SET_UNIVERSE_ACLS                                    STARTED 3:34:12 PM
------------------------------------------------------------------------------------------

------- !
!                                   : PUBLIC=>R,OPERATOR=>RCO,NETWORK_PUBLIC=>RC
------- defaults: !
!                                   : PUBLIC=>RW,NETWORK_PUBLIC=>RW
------- !commands.@
Context: !COMMANDS
ABBREVIATIONS                       : PUBLIC=>R,SYSTEM=>RC,OPERATOR=>RCO,NETWORK_PUBLIC=>R
ACCESS_LIST'SPEC'V(2)               : PUBLIC=>R,NETWORK_PUBLIC=>R
```

```
ACTION_UTILITIES'SPEC'V(2)          : PUBLIC=>R,NETWORK_PUBLIC=>R
ACTIVITY'SPEC'V(2)                  : PUBLIC=>R,NETWORK_PUBLIC=>R
ADA'SPEC'V(3)                       : PUBLIC=>R,NETWORK_PUBLIC=>R
ARCHIVE'SPEC'V(6)                   : PUBLIC=>R,NETWORK_PUBLIC=>R
CMVC'SPEC'V(9)                      : PUBLIC=>R,NETWORK_PUBLIC=>R
CMVC_ACCESS_CONTROL'SPEC'V(1)       : PUBLIC=>R,NETWORK_PUBLIC=>R
     ...

!TOOLS.XREF_UTILITY                 : PUBLIC=>R,SYSTEM=>R,OPERATOR=>RW,NETWORK_PUBLIC=>R
------- !tools.networking.@
Context: !TOOLS.NETWORKING
ARP_DEFINE'SPEC'V(1)                : PUBLIC=>R,NETWORK_PUBLIC=>R
ARP_SHOW'SPEC'V(1)                  : PUBLIC=>R,NETWORK_PUBLIC=>R
ARP_UNDEFINE'SPEC'V(1)              : PUBLIC=>R,NETWORK_PUBLIC=>R
BYTE_DEFS'SPEC'V(2)                 : PUBLIC=>R,NETWORK_PUBLIC=>R
BYTE_STRING_IO'SPEC'V(2)            : PUBLIC=>R,NETWORK_PUBLIC=>R
CMC                                 : NETWORK_PUBLIC=>RCO,MAILER=>R
CMC_TCP_IP_2_6_1'V(2)               : PUBLIC=>R,NETWORK_PUBLIC=>R
DEBUG                               : NETWORK_PUBLIC=>RCO
DUMP_FILE'V(31)                     : PUBLIC=>R,NETWORK_PUBLIC=>R
DUMP_FILE_IOP'V(29)                 : PUBLIC=>R,NETWORK_PUBLIC=>R
DUMP_FILE_TRACE'V(29)               : PUBLIC=>R,NETWORK_PUBLIC=>R
EXOS_8000_4_IA'V(2)                 : PUBLIC=>R,NETWORK_PUBLIC=>R
FILE_TRANSFER'SPEC'V(2)             : PUBLIC=>R,NETWORK_PUBLIC=>R
FTP_DEFS'SPEC'V(2)                  : PUBLIC=>R,NETWORK_PUBLIC=>R
FTP_NAME_MAP'SPEC'V(2)              : PUBLIC=>R,NETWORK_PUBLIC=>R
FTP_PRODUCT'SPEC'V(2)               : PUBLIC=>R,NETWORK_PUBLIC=>R
     ...
```

This output displays alphabetically the access rights of all worlds in the standard Environment, as well as all files and Ada units contained in these worlds.

After you execute this command, you may want to check for access-control problems. To do so, refer to the next subsection, where the Check_Universe_Acls command is described.

Checking Standard Environment ACLs

Users sometimes alter the access to certain files, Ada units, or worlds in ways that cause the system to lose some of its functionality. The Check_Universe_Acls command displays error messages indicating access-control problems in the standard Environment.

To execute this command, enter and promote Check_Universe_Acls in a command window. If this command finds no problems with the standard Environment's ACLs, it displays:

```
+++ No problems found.
```

If the command does find problems, it displays which groups or users have not been granted access rights to files, Ada units, or worlds in the standard Environment. For example, if a user is unable to send mail, executing the Check_Universe_Acls command might display error messages similar to the following:

```
*** Mailer does not have RW access to !MACHINE.TRANSFER
--- cannot send mail
*** Mailer does not have RW access to
!MACHINE.TRANSFER.CARRIER_MAP'V(3)
--- cannot send mail
*** Mailer does not have RW access to
!MACHINE.TRANSFER.CARRIER_QUEUES
--- cannot send mail
*** Mailer does not have RW access to !MACHINE.TRANSFER.DISTRIBUTE
--- cannot send mail
*** Mailer does not have RW access to
!MACHINE.TRANSFER.GLOBAL'V(3)
--- cannot send mail
*** Mailer does not have RW access to !MACHINE.TRANSFER.LOCAL'V(3)
--- cannot send mail
*** Mailer does not have RW access to
!MACHINE.TRANSFER.LOCAL_CARRIER
--- cannot send mail
*** Mailer does not have RW access to
!MACHINE.TRANSFER.SMTP_CARRIER
--- cannot send mail
*** network_public does not have R access to
!MACHINE.TRANSFER.DISTRIBUTE
--- Mail Distribute_Server cannot run as Network_Public.
++* 9 problems.
```

Limitations of Set- and Check_Universe_Acls

The Set_Universe_Acls and Check_Universe_Acls commands set the security level of systems at a site and check the access rights of files, Ada units, and worlds. These commands do not check or set the access rights of individual files, Ada units, or worlds that are not part of the standard Environment. Thus your system may have access-control problems that are not displayed as errors by the Check-_Universe_Acls command and that cannot be corrected by the Set_Universe_Acls command. In these cases, you need to use the individual access-control operations described in the "Procedures for Access Control" section.

Additionally, the Set_Universe_Acls and Check_Universe_Acls commands cannot be used when a system is so disabled that the commands themselves cannot be executed from a command window. If your system is experiencing access-control problems so severe that you cannot log into your system or execute Environment-level commands, contact Rational using the procedures described in Appendix D.

IMPLICATIONS OF ACCESS CONTROL

Access control interacts with various aspects of the Environment in ways that affect system management. These are described in the following subsections.

Name Resolution

When you specify an object name as a parameter value to a command, the Environment *resolves* the name by determining what object it references. The Environment cannot resolve a name to an object unless you have read access to the world that contains the object. This is especially important for naming expressions that contain wildcards.

For example, assume that you have read access to the current library, which contains worlds called Public and Private. Assume further that you have read access to the world called Public but not to the world called Private.

In this case, the following command lists the fully qualified name for Public, along with the fully qualified names for each object in Public:

```
Library.Resolve ("Public?")
```

In contrast, the following command lists only the fully qualified name for Private, without listing the names of any of the objects in Private:

```
Library.Resolve ("Private?")
```

Resolving an object name through a searchlist requires read access to the appropriate object and its containing world. Without the required access, the name appears undefined and the command does not execute.

Links, Compilation, and Execution

Adding a link from a world to an Ada unit requires read access to the unit and owner access to the world.

Promoting an Ada unit requires write access to that unit and read access to any unit it *withs*. Promoting a unit and its closure requires write access to every unit whose state must be changed. (A unit's closure is the set of units that it directly or indirectly *withs*.)

Demoting an Ada unit requires write access to the unit. However, no access rights are required to any of the units that *with* the unit being demoted.

Executing commands or programs from a command window requires read access to all units named in the window. Similarly, when executing the !Commands.Program.Run_Job, !Commands.Program.Run, and !Commands.Program.Create_Job commands, read access is required to any input files and write access is required to any output or error files that are named in the commands' Options parameter.

Networking

Operations provided by Rational Networking—TCP/IP are governed by access control. An FTP operation succeeds only if the *identity* specified in the operation's parameter list has the appropriate access to the appropriate objects.

When RPC servers are initialized on a machine, they are given the identity of the job or user that initiated them. For example, RPC servers initialized by !Machine-.Initialize are given the identity Network_Public. When a client program makes a request of a server, the server's identity is used to determine the client's access rights.

To ensure adequate access rights, a client can optionally specify a new identity for the server to assume for the duration of the request. To do this, the client must pass a valid username and password to the server, and the server must use the !Commands.Program.Change_Identity command to adopt the username as its new identity. Note that a server can have only one identity at a time. To process simultaneous requests specifying different identities, the server starts a separate job for each request. See the *Rational Networking—TCP/IP Reference Manual* for further information.

There may be times when you need to change the identity of the server without using a password. If the server is in privileged mode (or is changed to privileged mode), you can make the change without a password.

CMVC and Subsystems

When a project is managed by CMVC, access control must be managed by operations in package !Commands.Cmvc_Access_Control instead of those in package Access_List. Operations in package Cmvc_Access_Control automatically adjust access rights for the various subobjects within subsystems and views; furthermore, these operations can be used to grant *execution rights* for various commands in package !Commands.Cmvc. Refer to the Project Management (PM) book of the *Rational Environment Reference Manual* for further information about package Cmvc_Access_Control.

Archive Commands

When objects are saved using procedures from package Archive, the string form of each object's ACL is saved with it. You can restore objects with the saved ACLs; alternatively, you can specify new ACLs through options on the Restore operation. These options also permit restored objects to retain the ACLs of any objects they overwrite or to inherit their ACLs from the context into which they are restored. Note that when objects are restored into views, ACLs are always inherited, because package Cmvc_Access_Control must manage the ACLs of objects in views.

5

Maintaining System Efficiency

An integral part of the Rational Environment is a program called the system *daemon*, which periodically runs a set of special jobs that serve as system custodians. These jobs, called *clients*, maintain system efficiency by cleaning up the system's internal data structures, disposing of obsolete data, and reclaiming used storage space.

You can reschedule clients, run clients independently of the schedule, prevent a scheduled client from running, and display information about each client.

Three clients, Error_Log, Snapshot, and Disk, are explained in detail because they require more interaction with the system manager or have more impact on Environment users. These three clients are covered in separate sections later in this chapter.

The final part of this chapter contains sections on managing disk space and jobs and recovering when thresholds are reached.

THE DAEMON AND ITS CLIENTS

The different classes of *daemon clients* are described below:

- The *object managers* are special system clients that create, control, and modify the data structures for Ada programs, text files, directories, user accounts, and other Environment objects. When scheduled to run under the system daemon, object managers perform maintanence tasks. See Table 5-2 for a listing of the 16 object managers and the tasks they perform.
- The *major clients* are seven system clients that periodically check, record, and upgrade the state of the machine. Major clients include: the daemons Actions, Snapshot, and Disk, and the object managers Ada, DDB, Directory, and File.
- The *Daily client* is the group of system clients scheduled to run each day as part of the system daemon. These system clients perform regular maintenance tasks.
- The *Weekly client* is the group of system clients scheduled to run each week as part of the system daemon. These system clients perform regular maintenance tasks.

Table 5-1 shows which daemon clients call other clients or object managers and how often the clients run.

The subsections following the table describe the purpose of the clients.

Table 5-1 The Five Daemon Clients and What They Call

Client	Standard Schedule	Object Managers Called	Other Clients Called
Actions	2 hours	None	None
Snapshot	30 minutes	None	None
Disk	Daily	None	None
Daily	Daily	Ada, Directory, and File	Disk, Error_Log
Weekly	Weekly	Archived_Code, Code_Segment, Configuration, DDB, Group, Link, Null_Device, Pipe, Program-_Library, Session, Tape, Terminal, and User	None

The Object Managers

When users create and modify objects such as Ada programs, text files, and directories, or when objects such as user accounts and sessions are modified, changes are made to data structures within the Environment. Each data structure is managed by an *object manager*. Table 5-2 lists and describes the 16 Environment object managers and gives the standard schedule of when these object managers are run by the system daemon.

Table 5-2 Object Managers Run as Daemon Clients

Object Manager	Task	Standard Schedule
Ada	Manages the set of objects of class Ada. Keeps track of the parent/child relationships between the Ada units and their subunits. Stores ACL information for Ada units.	Daily
Archived_Code	Manages D1-style code views and load procs.	Weekly
Code_Segment	Manages R1000 Ada code segments.	Weekly
Configuration	Reclaims memory.	Weekly
DDB	Manages the dependency database, which maintains a record of the dependencies between Ada units. Removes obsolete dependencies from the database.	Weekly
Directory	Manages the relationship between objects in the directory system, including what objects are children of a directory, and default ACL information.	Daily
File	Manages the set of objects of class File. Stores ACL information for file objects.	Daily
Group	Manages groups.	Weekly
Link	Manages Ada *with* connections between worlds.	Weekly
Null_Device	Reclaims memory.	Weekly
Pipe	Is used by kernel (analogous to UNIX® pipes).	Weekly
Program_Library	Manages the Ada program library. (Before D3 release: associated with each world.)	Weekly

Table 5-2 Object Managers Run as Daemon Clients (continued)

Object Manager	Task	Standard Schedule
Session	Manages login sessions.	Weekly
Tape	Manages tape devices.	Weekly
Terminal	Manages terminal-port objects.	Weekly
User	Manages R1000 user accounts.	Weekly

The Major Clients

The seven *major clients* periodically check, record, and upgrade the state of the machine. The major clients are: the daemons Actions, Snapshot, and Disk, and the object managers Ada, DDB, Directory, and File (the object managers are discussed in the preceding subsection).

Just before running major clients, the system daemon sends warning messages to the operator's console. The system daemon also sends messages for Snapshot and Disk to the message windows of all system users.

See the "Displaying Information about Clients Using Daemon.Status" subsection for instructions on displaying the status of the major clients.

Actions Client

The *Actions client* manages the data structure that controls simultaneous access to objects. This data structure prevents inconsistencies when several users or jobs access an object simultaneously. In the standard Environment, the system daemon unobtrusively runs the Actions client every two hours.

Snapshot Client

The *Snapshot client* makes a record, or *snapshot*, of the current state of the Environment. Snapshots are important because, when the system boots, the Environment is restored to the state that was recorded by the most recent snapshot. Only objects that were committed before the last snapshot are preserved.

While the Snapshot client is running, it locks all committed objects for about 30 seconds—during this time, the objects are inaccessible to users.

See "The Snapshot Client" section for a more detailed description of this client, including techniques for running the Snapshot client and for controlling the content and display of messages associated with the Snapshot client.

Disk Client

Various object managers regularly mark unneeded data on the hard disks as obsolete. The *Disk client* searches for obsolete data, and marks the occupied space as free. This process is called *disk collection*. The Disk client normally runs each day.

For a description of the Disk client, see "The Disk Client" section later in this chapter.

The Daily Client

The Daily client represents the set of clients scheduled to be run each day by the system daemon:

- The *Disk client:* The Disk client searches for obsolete data and marks the occupied space as free. See "The Disk Client" section later in this chapter for details.
- The *Error_Log client:* The Error_Log client writes the system error log from temporary storage to a permanent, dated Environment file. See "The Error_Log Client" section later in this chapter for details.
- Several *object managers:*
  - Ada: Manages objects of the class Ada
  - Directory: Manages objects of the class Directory
  - File: Manages objects of the class File

  See "The Object Managers" subsection, above, for a description of all 16 Environment object managers.

By default, the system sends a warning message to all users 2 minutes before the Daily client runs.

The standard schedule for the Daily client specifies that its clients run in the order: Directory, File, Ada, Disk, and Error_Log, immediately after one another, once each day. (This order appears in the "Previous Time" field of the status display produced by the command: `Daemon.Status("Daily").`) The standard schedule runs the Daily client during the early morning hours to avoid causing performance problems during user hours.

See "The Daily Client's Schedule" subsection, later in this chapter, for factors to consider when setting the start time for the Daily client.

The Weekly Client

The Weekly client represents the set of clients scheduled to run once each week by the system daemon: Program_Library, Archived_Code, Pipe, Null_Device, Link, Code_Segment, Configuration, Terminal, Tape, Session, Group, User, and DDB. Note that these are all object managers. See Table 5-2 for a summary of what these object managers do, and see "The Weekly Client's Schedule" subsection for the time and order in which they run.

WHEN CLIENTS ARE RUN

Clients are run:

- By the system daemon according to the standard schedule
- As part of certain procedures
- As needed by the system
- By explicit request

These four methods are described below.

By the System Daemon according to the Standard Schedule

Daemon clients normally run automatically according to the daemon-schedule table. The scheduled times are selected to match normal system usage. For example, the standard schedule tells the daemon to run object managers during the early morning hours when system usage is at its lowest. This scheduling is necessary because object managers compact the objects that they manage, and an object undergoing compaction is not available to system users.

Under most circumstances, the standard client scheduling is the most appropriate. You can change or override the schedule, however, if you need to reschedule a client.

The daemon-schedule table is created each time the system boots. You can modify the standard schedule through commands in package Daemon or through the methods discussed for machine initialization. (See the "Tailoring the Standard Schedule" section, later in this chapter, and Appendix E. Also see package Daemon in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*)

Sometimes you may want a client to skip its next scheduled run. You might even want to prevent a client from running until you specifically allow it to run again. For a discussion of how to do this, see the "Preventing a Client from Running" subsection, later in this chapter.

Before you make any changes to the standard daemon-schedule table, Rational recommends that you consult a Rational technical representative.

As Part of Certain Procedures

In addition to running on a regularly scheduled basis, some clients (such as Snapshot) are run as part of certain Environment commands or procedures.

For example:

- The !Commands.Abbreviations.Do_Backup command runs the Snapshot client as part of the backup process.
- The !Commands.Operator.Shutdown command runs the Snapshot client before shutting down the system.
- The Disk client runs the Snapshot client as part of scheduled disk collection.

As Needed by the System

The system monitors system activity and, as needed, runs selected clients to maintain system efficiency. For example, the system runs the Disk client whenever the amount of free disk space falls below a preset threshold.

By Explicit Request

To run a client independently of the daemon's schedule table, use the !Commands.Daemon.Run command.

For example, if you want to run the Snapshot client immediately instead of waiting until the scheduled time, execute this command: `Daemon.Run("Snapshot")`.

After displaying the normal snapshot warning messages and waiting for the scheduled delay, the Snapshot client takes the requested snapshot.

Running a client with the !Commands.Daemon.Run command has no effect on the client's preset scheduling in the daemon schedule table.

Note: *When executing a client outside its normal schedule, consider the effect of that client on the system performance. See the following section, "The Standard Schedule and Its Effects," for a description of how different clients affect system performance.*

Note: *You can run the daemon clients only if you have operator capability. See the "Special Groups" subsection in Chapter 4 for a description of operator capability.*

THE STANDARD SCHEDULE AND ITS EFFECTS

Each client affects system response when it runs. Some clients, such as Error_Log or Link, have a negligible effect, because they run at a low level in the background for only a few seconds. In contrast, clients such as Ada, DDB, Directory, and File lock the objects they manage for 2 to 20 minutes, depending on the client. (When objects are locked, users cannot access them.) Furthermore, the Disk client can reduce system performance for a period of up to 60 minutes per disk drive. Also, when the Disk client starts, it can kill any system backup that is in progress.

Conversely, a heavy system load affects how quickly and efficiently a client can get its work done. When other jobs compete for resources, important background clients such as Disk can be delayed or take hours to run.

Therefore, it is important that the client scheduling:

- Allow clients to run often enough to keep the Environment in good working order
- Minimize the clients' effect on system response
- Minimize the system's effect on the clients' efficiency

Specifically, clients should run when system usage is low, and the Disk client *cannot* overlap system backups, because the backup will be killed (see the "System Backups" section in Chapter 7). The standard schedule achieves these goals by running the majority of the clients in 1.5 to 3 hours. During this time, the Environment is essentially unusable. This block of time is generally scheduled in the early morning, before the first users log in and after overnight batch jobs have completed. If necessary, you can reschedule the block around your users' schedules. (You may want to inform users of the client scheduling so that they can avoid logging in or scheduling batch jobs while the clients run.)

The following three subsections discuss the standard scheduling in greater detail.

Frequent Clients: Schedules for Actions and Snapshot

The Actions and Snapshot clients run frequently throughout the day. Their specific schedules are:

- The Actions client runs every 2 hours with no noticeable impact on system response.
- By default, the Snapshot client runs every 30 minutes and locks all objects for about 30 seconds, depending on how many objects have been modified since the last snapshot. (The more modified objects there are, the longer the Snapshot client runs.) How frequently the Snapshot client runs on your system can be changed. See the "When the Snapshot Client Runs" subsection for more information about scheduling this client.

The Weekly Client's Schedule

Under the standard schedule, the Weekly client runs several clients once a week. The Weekly client runs in the order indicated by the "Previous Time" field of the display produced by the Daemon.Status ("Weekly") command. This order is: Program_Library, Archived_Code, Pipe, Null_Device, Link, Code_Segment, Configuration, Terminal, Tape, Session, Group, User, and DDB.

By default, these clients are scheduled to begin running at 2:30 A.M. the day after the machine is booted and once per week thereafter, and they run for 5–10 minutes.

The Daily Client's Schedule

Under the standard schedule, the Daily client runs its object managers and clients in the order indicated by the "Previous Time" field of the display produced by the Daemon.Status ("Daily") command. This order is: Directory. File, Ada, Disk, and Error_Log. The system sends a warning message to all users 2 minutes before the Daily client runs. The Daily client is scheduled as a block of clients that run once a day during the early morning, and it runs for 1.5–3 hours. depending on the amount of data on the system.

Most objects are unavailable to users while the Daily client runs the object managers. The object managers run sequentially, one at a time The Disk client runs after the object managers finish, and the Error_Log client runs last. On a relatively new system, the Disk client takes 10–20 minutes per disk. As more objects are added to the system, the Disk client requires more time. While the Daily client runs, the system is not usable except from the operator's console.

The standard schedule for many systems starts the Daily client at 3:00 A.M. Consequently, the system is unusable until about 5:00 A.M. You can reschedule the Daily client if users regularly need to log in or run batch jobs during these times.

Note: *Only approximate times are given, because the running time of each client depends on the amount of data change since the client last ran. For example, if there are many changes to Ada units, the Ada client requires more time. The amount of time taken by each client varies from day to day.*

To determine the approximate amount of time each client requires, examine the daily system error logs in !Machine.Error_Logs or use the "System Log Information" tool from the *Rational Software Library Catalog.*

Rational recommends that you monitor the amount of time required by the Disk client, because Disk runs longer as the number of objects on the system increases.

TAILORING THE STANDARD SCHEDULE

The default standard client schedule is set by the system. You can modify the schedule, however, by making changes in machine initialization (see Appendix E). Machine initialization executes automatically as part of the boot process and can include changes to the default settings for the clients.

The disk-collection thresholds also are recalculated automatically each time the system is rebooted. These thresholds are discussed later in the "Automatic Disk-Collection Thresholds" subsection.

You can also make temporary changes through commands in package Daemon.

You should consider modifying *only* these aspects of the standard schedule:

- What time the Daily client should run. Start the Daily client after the last user logs out or after overnight batch jobs have finished, and leave enough time for the Daily client to complete before users log in again.
- How soon and how often the Snapshot client should run. A snapshot should be taken often enough so that minimal work is lost if the system fails but not so often that users' work is constantly interrupted. You may also want to change the time settings for the warning and start messages.

To modify the standard schedule, you must make the appropriate changes in machine initialization. See the "Rescheduling the Daily Client" subsection, below, and Appendix E for more information.

Hints for Specifying Duration Parameters in Commands

Commands for scheduling include parameters that require values of type Duration. Type Duration is defined in package Standard, and it represents the decimal notation for seconds. For example, a duration of 3600.0 represents 3,600 seconds, or one hour.

For convenience and fewer errors, you may choose to specify parameters of type Duration by using resources from package !Tools.Time_Utilities.

You can use the Minute, Hour, and Day constants from package !Tools.Time_Utilities, because they are of the Duration type. For example, if you want to specify a duration of 2 hours, instead of specifying a duration of:

```
7200.0 seconds
```

you can specify a duration of:

```
2 * Time_Utilities.Hours
```

The Time_Utilities.Duration_Until_Next function is useful because it returns the number of seconds between the time of execution and a specified time of day. This allows the parameter value to be independent of the time at which the Duration_Until_Next function is executed.

For example, consider a parameter of type Duration that represents the amount of time until some event should take place. Suppose it is now 7:37 P.M., and you want the event to occur at 2:15 A.M. Instead of calculating the amount of time between now and the desired event (which will be different by the time you calculate it!), use the Duration_Until_Next function to do this calculation for you

—simply specify the clock time at which you want the event to occur, using the following 24-hour time format:

```
hh,mm,ss
```

where:

- *hh* represents hours in the range 0–23
- *mm* represents minutes in the range 0–59
- *ss* represents seconds in the range 0–59

For example, to specify the clock time of 2:15 A.M., use:

```
Time_Utilities.Duration_Until_Next (02,15,0)
```

when specifying any parameter of type Duration.

Rescheduling the Snapshot Client

Given the volume of work done in half an hour on a particular machine, you may decide that snapshots should be taken more frequently. (See the "When the Snapshot Client Runs" subsection, later in this chapter, for more information about the standard schedule for snapshots.) The following example shows how to make a schedule change for the Snapshot client through machine initialization (see Appendix E).

To schedule the Snapshot client to run every 15 minutes:

1. Create or open for editing a text file called !Machine.Initialization. Local.Snapshot_Changes_Start.
2. Edit the file to include the following lines:

```
--|Procedure_Name Daemon.Schedule
--|Procedure_Context !Commands
--|Parameters Client => "Snapshot",
--|Parameters Interval => 15 * Time_Utilities.Minute,
--|Parameters First_Run => 15 * Time_Utilities.Minute)
```

3. Commit the Snapshot_Changes_Start file by pressing [Commit] or [Promote].
4. Execute the !Machine.Initialization.Start command with the following parameters:

```
Start (Procedures_To_Run => "Snapshot_Changes_Start"
       Effort_Only       => False)
```

After the Start procedure runs, you should see a confirmation message similar to:

```
92/08/09 17:09:35 --- At 8/09/92 17:24 and every 15:00.000 run
Snapshot.
```

Note: *This new schedule remains in effect each time the machine reboots, unless the !Machine.Initialization.Local.Snapshot_Changes_Start file is modified or deleted.*

Rescheduling the Daily Client

Under the standard scheduling, the system is unusable from 3:00 A.M. until about 5:00 A.M. while the Daily client runs. If users regularly log in around 4:00 A.M., however, you may want to start the Daily client earlier—at 2:15 A.M., for example.

To reschedule the Daily client:

1. Create or open for editing a text file called !Machine.Initialization.Local.Daily-_Client_Changes_Start.
2. Edit the file to include the following lines:

```
--|Procedure_Name Daemon.Schedule
--|Procedure_Context !Commands
--|Parameters Client => "Daily",
--|Parameters Interval => Time_Utilities.Day,
--|Parameters First_Run =>
                    Time_Utilities.Duration_Until_Next (2,15))
```

   Note: *The First_Run parameter is given the Duration_Until_Next function as its argument so that we can specify the duration in terms of the time of day we want the Daily client to run (in this case, 2:15 A.M.), as opposed to the amount of time remaining until we want the Daily client to run. See "Hints for Specifying Duration Parameters in Commands," above, for a complete description of this and other functions returning type Duration.*

3. Commit the Daily_Client_Changes_Start file by pressing [Commit] or [Promote].
4. Execute the !Machine.Initialization.Start command with the following parameters:

```
Start (Procedures_To_Run => "Daily_Clients_Changes_Start",
       Effort_Only       => False)
```

   After the Start procedure runs, you should see a confirmation message similar to:

```
92/08/09 18:15:11 --- At 8/10/92 02:15 and every 1/00:00 run
                  ... Directory.
92/08/09 18:15:11 --- At 8/10/92 02:15 and every 1/00:00 run
                  ... File.
92/08/09 18:15:11 --- At 8/10/92 02:15 and every 1/00:00 run
                  ... Ada.
92/08/09 18:15:11 --- At 8/10/92 02:15 and every 1/00:00 run
                  ... Disk.
92/08/09 18:15:11 --- At 8/10/92 02:15 and every 1/00:00 run
                  ... Error_Log.
```

Note: *This new schedule remains in effect each time the machine reboots, unless the !Machine.Initialization.Local.Snapshot_Changes_Start file is modified or deleted.*

Rescheduling Clients Temporarily

When you reschedule clients by modifying files or procedures in the !Machine-.Initialization.[Local,Site,Rational] worlds, the changed schedule stays in effect indefinitely because it is reset every time the system boots. It is also possible to change the scheduling temporarily so that the changes stay in effect only until the next time the system boots. Then, at boot time, the standard scheduling takes effect again. For example, if you know that users will be working unusual shifts for one day, you can reschedule the clients around the temporary work shifts.

There are two general methods for rescheduling clients temporarily:

- Use the methods discussed for making changes through machine-initialization @_Start files—except place the @_Start files in some world other than !Machine-

.Initialization.[Local, Site, Rational]. Then specify that world as the argument to the Local_Initialization_Directory parameter when executing the Start procedure. For example, to set up a temporary Daily client schedule that can be used as needed, follow the steps described in the "Rescheduling the Daily Client" subsection, above, except:

1. For step 1, create or open for editing a text file called Daily_Client_Changes-_Start in some world *other than !Machine.Initialization.[Local, Site, Rational].* For example, you can create a world called !Machine.Initialization.Temporary to contain @_Start files intended for temporary schedule changes.
2. For step 4, execute the !Machine.Initialization.Start command with the following parameters:

```
Start (Local_Initialization_Directory =>
                          "!Machine.Initialization.Temporary",
       Procedures_To_Run => "Daily_Clients_Changes_Start",
       Effort_Only       => False)
```

   substituting the pathname to the world containing your temporary schedule changes in the Local_Initialization_Directory parameter.

You may need to set up the world and @_Start files for temporary schedules only once—after that, you can simply execute the Start command as in step 2.

- Execute one or more !Commands.Daemon.Schedule commands directly, filling in the parameters appropriately.

Note: *These temporary changes remain in effect until the machine reboots, or you explicitly change the schedule back to its original setting. You can use the same method for resetting the machine as you used to set the temporary changes, if you know the original settings. Alternatively, you can include @_Start files or initialization procedures in the !Machine.Initialization.Local world to reset defaults.*

Preventing a Client from Running

Note: *There is no way to cancel a client if it is currently running; you can prevent it from running only by suspending its schedule* ***before*** *it starts to run.*

Occasionally, you must suppress, or *quiesce*, a client from running at its scheduled time so that:

- Important user jobs can continue without interruption (such as a backup or installation)
- You can recover data before a client obliterates it *forever*

To prevent a client from running until a specified amount of time after the currently scheduled time, use the !Commands.Daemon.Quiesce command.

For example, to prevent the Daily client from running for one additional day (the default additional delay), execute the command:

```
Daemon.Quiesce ("Daily")
```

For further information about the Quiesce command, see procedure !Commands-.Daemon.Quiesce, described in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Suppressing for a Specified Time

By default, a quiesced client runs again 24 hours after the currently scheduled next run. Because this is not desirable for certain clients (Snapshot, for example), you may want to change the default.

For example, to prevent the Error_Log client from running until 4 hours after the currently scheduled time, execute the following command:

```
Daemon.Quiesce ("Error_Log", 4 * Time_Utilities.Hours)
```

The Error_Log client is now scheduled to run 4 hours later than its previously scheduled time.

Suppressing Indefinitely

To quiesce a client until you explicity reschedule it, execute the !Commands.Daemon.Quiesce command with the parameter value Duration'Last. For example:

```
Daemon.Quiesce ("Snapshot", Duration'Last)
```

When you want to restart an indefinitely quiesced client, reschedule it with either the machine-initialization method or the Daemon.Schedule command, as described in the previous subsections on rescheduling clients.

Note: *A system reboot also reschedules an indefinitely quiesced client.*

Running Clients Independently of the Schedule

Clients on the standard schedule normally run often enough to maintain system efficiency. On occasion, however, you may need to run a client independently of its schedule. The !Commands.Daemon.Run command allows you to run a client as needed in addition to the client's scheduled runs. Running a client using !Commands.Daemon.Run does not interfere with the client's preset schedule, but it can slow system response considerably, so avoid using !Commands.Daemon.Run when the system load is high.

The Disk client and the Snapshot client are usually the only two clients you may need to run by explicit request. See "The Snapshot Client" and "The Disk Client," below, for more information about running these clients independently of the schedule.

To run a client by explicit request, execute the !Commands.Daemon.Run command, including the name of the client you want to run.

For example, to run the Error_Log client:

1. Execute the command:

   ```
   Daemon.Run ("Error_Log")
   ```

   The Error_Log client runs immediately.
2. Press [Control] [G] to disconnect from the job; the client runs in the background.

OBTAINING INFORMATION ABOUT CLIENTS

Information about clients is available from three sources:

- From the messages sent by clients as they run
- From the display generated by the !Commands.Daemon.Status command
- From the Environment error-log files

Client messages are kept in Environment error-log files, where you can inspect them to find out how long each client takes to run. Some clients display messages on the operator's console and on user terminals. The !Commands.Daemon.Status display shows both scheduling information and, for the object managers, information about the size of the managed data structures.

Messages Generated by Clients

When clients start and finish, they issue messages to both the console and the error log. For example, when the Ada client runs, it produces a pair of messages similar to the following:

```
03:08:31 +++ Ada Started
03:08:37 +++ Ada Completed old = 338; new = 257 (pages)
```

By comparing times in the starting and finishing messages, you can tell how long the client took to run. In the example above, the Ada client ran for 6 seconds.

Furthermore, if the client is an object manager, its finishing message reports the size of the object manager's data structure, in pages, before and after the client ran. In this example, the Ada object manager's data structure was 338 pages before the Ada client ran and 257 pages after.

Under the default message handling, the messages for all clients are displayed on the operator's console as the clients run. By watching the operator's console, you can monitor the clients' progress as they run.

In addition to the operator's console display, the messages for all clients are written onto disk and then transferred once a day to error-log files in the Environment. For more information about the error log, see the following section, "The Error_Log Client."

Note: *All messages from the Disk client are written into these logs, even though not all of the messages are displayed on the operator's console when they are issued.*

Displaying Information about Clients Using Daemon.Status

You can use the !Commands.Daemon.Status command to display information about a single client, the major clients, or all clients.

Use !Commands.Daemon.Status when you want to:

- Find out when a client last ran and when it runs again
- Track an object manager's effect on an object's size
- Find out which phase of disk collection the Disk client is in
- Find out how many disks scheduled for collection have actually been collected

For Single Clients

To display information about a single client, execute the Daemon.Status command including the name of the desired client.

For example, to display information about the Disk client:

```
Daemon.Status ("Disk");
```

The !Commands.Daemon.Status command displays information about the specified client similar to:

```
Client     Next Time      Previous Time  Interval  Size   Post   Pre
====== ============== ============== ======== ====== ====== ======
Disk   08/09/92 04:00 08/08/92 06:08 01/00:00 745770 654461 923935
Vol 1                 08/08/92 04:53          218340 167547 310214
Vol 2                 08/08/92 05:33          289956 266614 341871
Vol 3                 08/08/92 06:08          237474 220300 271850
```

The description of this display is:

- *Client:* The client's name.
- *Next Time:* The time of the client's next scheduled run.
- *Previous Time:* The actual time of the client's previous run. Note that the actual time a client runs may not match the scheduled time; the client may run later because of system load, how long preceding clients run, and so on.
- *Interval:* The interval of time between scheduled runs. The format for expressing intervals is as follows:
  - *mm:ss.ff* indicates the number of minutes, seconds, and decimal fractions of seconds between runs. For example, the Actions client runs every 30 minutes, or 30:00.00.
  - *hh:mm:ss* indicates the number of hours, minutes, and seconds between runs. For example, a client that runs every hour would display `01:00:00`.
  - *dd/hh:mm* indicates the number of days, hours, and minutes between runs. For example, a client that runs once a day would display `1/00:00`.
- *Size:* The current size, in pages, of the client's data structure.
- *Post:* The size, in pages, of the client's data structure after the last run.
- *Pre:* The size, in pages, of the client's data structure just before the last run.

Note: *Disk-client information: An asterisk appearing after a volume number while the Disk client is running indicates that the disk is scheduled for collection, and the Disk client has not yet performed disk collection on that disk. While the Disk client runs, it displays additional information described later in this chapter.*

For Major Clients

To display information about all major clients, execute the !Commands.Daemon-.Status command with its default parameter `"*"`:

```
Daemon.Status ("*")
```

A typical display for the major clients is:

```
Client          Next Time          Previous Time      Interval     Size      Post      Pre
=========  ================  ================  ========  ======  ======  ======
Actions    08/08/92  11:53   08/08/92  11:23   30:00.00      55      55      55
Ada        08/09/92  04:40   08/08/92  05:03   01/00:00    2073    2010    2425
DDB        08/08/92  04:15   08/08/92  04:19   01/00:00   11213   10899   10899
Directory  08/08/92  04:30   08/08/92  04:36   01/00:00    5775    5698    6423
Disk       08/09/92  05:00   08/08/92  06:24   01/00:00  339000  311000  351000
File       08/08/92  04:45   08/08/92  05:14   01/00:00    1103     980    1219
Snapshot   08/08/92  12:08   08/08/92  11:10   01:00:00
```

Note: *The Actions client's data structure does not change size.*

For All Clients

To display the status of all clients, enter a null string as the parameter to !Commands.Daemon.Status:

```
Daemon.Status ("")
```

Changing the Warning Interval Given to Users

To increase or decrease the warning interval given to users before the Daily clients begin, use the !Commands.Daemon.Warning_Interval command. For example, to set the warning interval to 300.0 seconds (5 minutes), execute this command:

```
Daemon.Warning_Interval (300.0)
```

THE ERROR_LOG CLIENT

The Error_Log client writes the system error log from temporary storage to a permanent, dated Environment file.

Setting the Destination for Error Messages

System errors and other messages can be directed to the operator's console, the *temporary-storage* error log on disk, or both with the Daemon.Set_Log_Threshold command (see package Daemon in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* for details).

Error-Log Files

Normally, one Environment error-log file is created per day, because the Error_Log client runs once a day under the standard schedule. The name of each error-log file shows the date and time it was created. For example, the file created on August 1, 1992, at 4:00 A.M. is named Log_92_08_01_At_04_00_01. The Error_Log client is normally the last client to run each day. Thus, the Environment error-log file that is created on a given day contains the messages that were generated on the previous day.

Accessing Error Messages

Messages in the temporary-storage error log are not accessible from the Environment until the Error_Log client copies the temporary-storage error log into the !Machine.Error_Logs file.

Messages are accessible, however, if:

- You run the Show_Error_Log command from the EEDB `Kernel:` prompt:

  ```
  ====>> Elaborator Database <<====
  Kernel: Show_Error_Log
  ```

  This command immediately displays the temporary-storage error log on the console.

- You run the Error_Log client by executing the Daemon.Run command from an Environment command window:

  ```
  Daemon.Run ("Error_Log");
  ```

  This command immediately writes the temporary-storage error log to a permanent, dated file in !Machine.Error_Logs.

THE SNAPSHOT CLIENT

The *Snapshot client* makes a record, or *snapshot*, of the current state of the Environment. Snapshots are important because, when the system boots, the Environment is restored to the state that was recorded by the most recent snapshot. Only objects that were committed before the last snapshot are preserved.

While the Snapshot client is running, it locks all objects for about 30 seconds. (This time varies between systems: the actual time during which objects remain locked depends on how many objects have been modified since the last snapshot, and on how much each object has been modified. The greater the number of objects that have been modified and the greater their modification, the longer the Snapshot client runs.)

The Snapshot client can be started by:

- The daemon: By default, the daemon runs the Snapshot client once every 30 minutes, unless you change this interval.
- The system: The system automatically runs the Snapshot client when you enter the !Commands.Operator.Shutdown command. Snapshots are not taken automatically when you stop the system with the [Break] key or by cutting power.
- Other clients: The Snapshot client is automatically run before the Disk client or a system backup is started. (Unlike regular snapshots, this snapshot is permanently retained.)
- The system manager: You can run the Snapshot client any time you feel you need to save a record of the current state of the system Environment.

The Snapshot client is scheduled and run like any other client. The Snapshot client (and the Disk client) differ from other clients, however, in two important ways:

- Snapshots issue warnings and informative messages as they run; the message mechanism includes settings that you can change.
- On certain occasions, snapshots must be run independently of the schedule.

Changing Snapshot Message Settings

By default, a snapshot sends the following three kinds of messages to users' message windows:

- A warning message 2 minutes before the snapshot begins
- A start message when the snapshot begins
- A finish message when the snapshot completes

To change these defaults, modify machine initialization (see Appendix E).

You can specify a different interval for the warning message, and you can suppress the display of any or all of these messages.

Snapshot Warning Message

When the Snapshot client runs, committed objects are inaccessible, and open objects cannot be committed. Therefore, the Snapshot client sends a warning message to give users time to access or commit objects before the snapshot begins.

By default, a warning message such as the following is sent 2 minutes before the snapshot begins:

```
from System: 2:29:23 PM;  Snapshot will start in 20 seconds
```

The warning message displays both the current time and the number of minutes until the snapshot begins.

To set a longer or shorter period of time between the warning message and the snapshot, change the snapshot interval. Permanent changes should be made to machine initialization (see Appendix E for more information). To change the snapshot interval temporarily, execute the following command:

```
Daemon.Snapshot_Warning_Message (n.0)
```

where *n* expresses the desired interval as a number of seconds.

For example, to set the new interval to 1.5 minutes, execute:

```
Daemon.Snapshot_Warning_Message (90.0)
```

The interval parameter is of Duration type, so you can specify a value by using the Minute constant from package !Tools.Time_Utilities. For example, to set the new interval to 5 minutes, execute:

```
Daemon.Snapshot_Warning_Message (5 * Time_Utilities.Minute)
```

Setting the interval to 0.0 suppresses the snapshot warning message entirely.

When the system boots, the snapshot warning interval resets to the default value (120.0 seconds), unless permanent changes are executed through machine initialization.

Snapshot Start and Finish Messages

The snapshot start message alerts users that they can expect diminished system response for a brief period, during which time they will not be able to access committed objects or commit open objects. The snapshot finish message alerts users that they can continue normal operations.

Typical start and finish messages are:

```
from System: 02:31:55 PM;  Snapshot has started
from System: 02:34:34 PM;  Snapshot has completed
```

Either or both of these messages can be suppressed. To prevent the start message from appearing when a snapshot starts, execute the following command:

```
Daemon.Snapshot_Start_Message (False)
```

To prevent the finish message from appearing when a snapshot completes, execute the following command:

```
Daemon.Snapshot_Finish_Message (False)
```

To resume these messages, execute the appropriate command with the parameter `True`.

By default the Snapshot client sends only the finish message, and booting the system resets this default. Therefore, if you want to change the start and/or finish message settings permanently, you must change machine initialization (see Appendix E for more information).

Displaying the Snapshot Message Settings

To determine the current snapshot message settings, execute this command:

```
Daemon.Show_Snapshot_Settings
```

A typical display is:

```
Snapshot Settings -- Interval - 02:00.000, Start = FALSE,
                                           Finish = TRUE
```

In this display:

- *Interval:* Indicates the number of minutes, seconds, and milliseconds between the snapshot warning message and the beginning of the snapshot.
- *Start:* Indicates whether a start message is displayed (True) or not (False).
- *Finish:* Indicates whether a finish message is displayed (True) or not (False).

When the Snapshot Client Runs

Like other clients, the Snapshot client is executed regularly by the system daemon. The default schedule is once every 30 minutes. Snapshots can be scheduled differently (see "Tailoring the Standard Schedule," earlier in this chapter).

Your Snapshot schedule should take into account the following four points:

- Infrequent snapshots interrupt users less often, but they are more disruptive when they do run because they take longer.
- Frequent snapshots ensure an up-to-date record of the system state. An up-to-date snapshot prevents loss of work during a system failure, because the system boots to the state recorded by the most recent snapshot.
- Frequent snapshots interrupt users as they work and could interfere with their productivity. Users cannot access objects during a snapshot.
- Frequent snapshots use more disk space.

Snapshots are taken automatically when you execute the Operator.Shutdown command. Snapshots are not taken automatically, however, when you halt the system with the [Break] key or the Emergency Off button, as described below in "Taking a Snapshot by Explicit Request."

Finally, *retained* snapshots are taken during disk collection and system backups.

Taking a Snapshot by Explicit Request

Occasionally, you must take a snapshot by explicit request. In particular, Rational recommends that you take a snapshot from the `EEDB:` prompt before halting the system with the [Break] key or with the Emergency Off button. Without a snapshot, work may be lost when the system reboots—specifically, objects that are uncommitted since the last snapshot. See the "Emergency Power Off" and "Deliberate

Environment Crash Using the Operator's Console [Break] Key" sections in Chapter 2 for more information about taking a snapshot from the EEDB: prompt.

Under some circumstances, you may not be able to take a snapshot—for example, if your system is hung. To preserve work committed since the last snapshot, contact a Rational support representative before halting the system. Rational may suggest steps that record the system state. See Appendix D for information on obtaining customer support.

You can also take a snapshot after committing changes to an important object (for example, an Ada program, a text file, or a port configuration). Taking a snapshot ensures that this object will not be lost if the system fails. Consider waiting until the next scheduled snapshot, however, since snapshots affect all logged-in users.

Note: *Operator capability is required to execute this command (see the "Special Groups" subsection in Chapter 4 for a description of operator capability).*

To take a snapshot, execute the following command:

```
Daemon.Run ("Snapshot")
```

When the command executes, the snapshot warning message is sent to all users. Then, after the warning interval has elapsed, the snapshot begins.

To take a snapshot immediately, without waiting for the warning interval to elapse, set the warning interval to 0.0, take the snapshot, and reset the warning interval to its previous value. For example, if the normal warning interval is 2 minutes, execute the following commands:

```
Daemon.Snapshot_Warning_Message (0.0)
Daemon.Run ("Snapshot")
Daemon.Snapshot_Warning_Message (120.0)
```

THE DISK CLIENT

This section is an overview of how the Disk client works. See the following section, "Managing Disk Space," for a description of commands that you can use to manage the Disk client.

The Disk client is responsible for collecting and disposing of obsolete data that accumulates on disks as objects are expunged and modified. Running the Disk client is also called *disk collection.* Under the standard schedule, disk collection is performed once a day, after the other Daily clients (except Error_Log) have run and before user jobs have begun. Note that, by default, when the Disk client starts, it kills any system backup that is in progress. If this is unacceptable, the Disk client can be set to wait for the system backup to finish.

The Disk client can collect space on only one disk at a time, so the Daily client instructs the Disk client to collect space on each disk, one after the other.

The Disk client can be started by:

- The daemon: By default, the daemon runs the Disk client once a day as one of its daily system-maintenance tasks, unless you have modified the Disk client parameters in the daemon-schedule table.
- The system: The system runs the Disk client any time the amount of data on the hard disks exceeds preset *disk-collection thresholds.*
- The system manager: You can run the Disk client any time you feel you need to reclaim disk space containing obsolete data.

Disk Volumes

Your R1000 can have one to four disks, depending on its configuration. Many of the commands that control disk collection can be applied to a single disk or to all of the disks. Each disk has a volume number (1, 2, 3, or 4) that you can use to single out an individual disk. In many commands, you can use the number 0 (in place of a single disk number) to indicate the entire set of disks.

Operations That Create Obsolete Data

Various operations within the Environment create obsolete data. The space consumed by obsolete data is reclaimed by the Disk client. For example, the following activities generate obsolete data:

- Using the Environment itself (for example, resolving directory names, killing jobs, and forcing users to log off).
- Committing a text file (this creates a new version of the text file, making the oldest retained version obsolete).
- Editing an Ada unit (this creates a new version of the unit, making the oldest retained version obsolete).
- Promoting and demoting units (for example, promoting a 100-kilobyte unit from source to installed state creates 100 to 300 kilobytes of obsolete data).
- Elaborating a large program whose size is n megabytes on a system with medium-term scheduler working-set limits of l megabytes will generate about $n - l$ megabytes of obsolete data.
- Coding a unit of any size converts the unit's previous code segments to obsolete data. This type of obsolete data space cannot be reclaimed until the next time the system is booted and disk collection is run.

Certain types of declarations consume large amounts of space that become obsolete data when the program terminates. An example of this type of declaration is:

```
subtype Length_Type is Positive range 1 .. 2 ** 16;
type Bounded_String (Static_Len : Length_Type := 80) is
    record
        Dynamic_Len : Length_Type;
        Chars       : String (1 .. Static_Len);
    end record;
Message_Table : Array (1 .. 250) of Bounded_String;
```

During elaboration, the declaration will initialize 16 megabytes of data stack, because the discriminant of type Bounded_String has a default initialization expression. Thus, although Chars'Length for each element in the Message_Table array is only 80 characters, 2 ** 16 characters are actually allocated to each element.

Initiating the Disk Client

The Disk client can be initiated in several ways:

- As discussed above, the scheduler initiates the Disk client at predetermined times, normally once a day. At this time, disk collection is performed on all disks.

- The system itself can initiate the Disk client for a particular disk (or disks) if disk space gets below a predetermined level, called the *Start_Collection threshold.*
- You can initiate disk collection explicitly with the !Commands.Daemon.Run ("Disk") or the !Commands.Daemon.Collect command. Disk-collection priorities are described below in "Automatic Disk-Collection Thresholds."

Daily disk collection typically is sufficient. Occasionally, however, additional disk collection is required. For example, a heavily used disk may accumulate too much obsolete data to be handled by daily disk collection. In some instances, what is required is not additional disk collection but rather raising the job priority of scheduled disk collection.

In daily disk collection, the space on each disk is reclaimed one disk at a time, from most full to least full. If the system initiates disk collection and if several disks reach the Start_Collection threshold at about the same time, the Disk client collects space on each one individually. The disk that reaches the Start_Collection threshold first is the first collected.

The disk-collection thresholds are described below.

Automatic Disk-Collection Thresholds

The system uses *disk-collection thresholds* to determine when disk collection is required. These thresholds are set automatically at boot time and can be customized through machine initialization (see Appendix E).

The thresholds are:

- *Start_Collection:* The system initiates disk collection on a volume as soon as the free space left on it is at or below the Start_Collection threshold.
- *Raise_Priority:* The system notifies the operator when the Raise_Priority threshold is reached. If the amount of free disk space on any volume falls below the Raise_Priority threshold, the system increases the priority of disk collection for that volume.
- *Stop_Jobs:* If free disk space continues to shrink until it reaches the Stop_Jobs threshold, a warning is sent to the operator's console and all user jobs are stopped to allow disk collection to complete. User jobs resume when disk collection finishes.

  Note: *It is important to allow the Environment to resume jobs on its own. Attempting to shut down and reboot the system uses additional disk space.*

  Sometimes, however, the system quickly reaches the Stop_Jobs threshold after disk collection has just completed. In this case, you may want to follow the procedures described in the "Recovering When the Stop_Jobs Threshold Is Reached" section, later in this chapter.
- *Suspend_System:* This threshold suspends all system activity. Disk collection stops. If this threshold is reached, you can use the procedure described in the "Recovering When the Suspend_System Threshold Is Reached" section, later in this chapter, to lower the value for this threshold. *Note that this situation should be extremely rare if other procedures in this chapter are followed properly.*

These thresholds are ordered. Thus, the Start_Collection percentage must be greater than the Raise_Priority percentage, which must be greater than the Stop_Jobs percentage, which must be greater than the Suspend_System percentage.

Notification of Users

The system notifies users about the state of disk collection when:

- Disk collection is about to start because the Start_Collection threshold is about to be reached
- Disk collection starts
- Disk-collection priority changes
- Disk collection finishes

You can change the notification interval to give users more (or less) warning before disk collection begins.

Phases of Disk Collection

The Disk client can be in any one of the following six phases:

- Idle
- Waiting_For_Backup_To_Finish
- Taking_Snapshot
- Deleting_Segments
- Traversing_Virtual_Memory
- Reclaiming_Blocks

These phases are ordered. Thus, the Disk client starts out in the Idle phase, waiting to be initiated. Once it is initiated, if there is a backup running and if your system is configured to delay disk collection during the backup command, the Disk client may wait for the backup to finish before it begins running (by default, the Disk client kills the backup). Next, it takes a special kind of snapshot, called a *retained snapshot.* Then it deletes segments, traverses virtual memory, and finally reclaims blocks. These phases are described in detail below.

If you execute the Daemon.Status ("Disk") command while disk collection is running, the display indicates which phase of disk collection the Disk client is in. The display also displays an asterisk after volume numbers that are scheduled to have disk collection run on them during the current collection process but whose space has not yet been collected.

Idle Phase

When the Disk client is idle, it is waiting to be initiated explicitly by a system task, the scheduler, or you. In this state, available disk space is greater than the Start_Collection threshold.

Waiting_For_Backup_To_Finish Phase

Backups and the Disk client cannot be run at the same time. Thus, the backup either must be terminated or must be allowed to complete before disk collection can begin. Unless your Rational technical representative has changed this, *backup kill mode* is enabled on your system. (See the "System-Backup Schedule" subsection in Chapter 7 for a description of the !Tools.Disk_Daemon.Set_Backup_Killing command.)

When backup kill mode is enabled, a backup that is running at the time disk collection begins will be terminated. If backup kill mode is disabled, the backup will continue to run until it is complete, with the Disk client in the Waiting_For_Backup_To_Finish phase until the backup is complete. When the system boots, backup kill mode is enabled automatically. Thus, unless you or your Rational technical representative has changed this, you will never see the Disk client in this phase.

Taking_Snapshot Phase

When the system enters the Taking_Snapshot phase, it is taking a retained snapshot that is used only by backups and the Disk client. (Only one snapshot can be taken and retained at a time, which is why both the Disk client and backups cannot run at the same time; both require this retained snapshot to be taken at their initiation.)

Deleting_Segments Phase

Once the retained snapshot is taken, the Disk client enters the Deleting_Segments phase. When the Disk client deletes segments, it removes objects that have been expunged by users. When an object is expunged, the disk space allocated to it is not returned to the available disk space until disk collection has completed.

Traversing_Virtual_Memory Phase

Next, the Disk client enters a phase in which it traverses virtual memory, going through every kernel data structure to find all reachable blocks. Unreachable blocks have obsolete data in them, and those blocks will be reclaimed.

Reclaiming_Blocks Phase

Finally, once all unreachable blocks are identified, they are reclaimed as usable disk space by returning them to the pool of available blocks.

Disk-Collection Priorities

As described above, you can initiate the Disk client by running the !Commands-.Daemon.Run ("Disk") or the !Commands.Daemon.Collect command, or it can be initiated automatically at scheduled intervals or by the system when the Start_Collection threshold is reached for a volume.

The Disk client can be run at different levels of priority. The Daily client runs disk collection at the highest priority. If you initiate disk collection explicitly using Daemon.Collect, you can specify any priority, whereas the Daemon.Run ("Disk") procedure runs disk collection at the lowest priority. If the system initiates the Disk client because a disk has reached the Start_Collection threshold, it determines a priority based on the amount of free space on a volume and the number of volumes that have reached the Start_Collection and Raise_Priority thresholds. The Raise_Priority threshold means that the situation is getting more critical: more disks are getting full, or the disk that has reached the Start_Collection threshold has less percentage of space free on it.

The priority affects other jobs running on the system. The priority of the Disk client ranges from using only spare cycles not used by other jobs to preempting all other system activity except that issued from the console.

For automatic disk collection, a policy function specifies the minimum allowable collection priority based on the number of volumes that have reached the Start-_Collection threshold and the Raise_Priority threshold. This default policy is:

- If no volumes are past the Raise_Priority threshold, the priority policy is based on the number of volumes that have passed the Start_Collection threshold.
- If a single volume has passed the threshold, the priority is –1.
- If two volumes have passed the threshold, the priority is 0.
- If three volumes have passed the threshold, the priority is 1.
- Any other set of conditions requiring additional disk-collection resources results in a collection priority of 2.

If you determine that this collection priority is not high enough, you can raise the priority (see "Changing the Priority of In-Progress Disk Collection," below).

The effect of this policy is:

- When a single volume has passed the collection threshold, the Disk client will run using only spare cycles.
- When more than one volume requires disk collection, *backing off* (using only spare cycles) is terminated.
- When the Raise_Priority threshold is reached, the disk-collection priority is raised so that the collection preempts most background jobs.

Your Rational technical representative can change this policy for you, if necessary.

The system impact based on the various levels of priority is as follows:

- Priority of –1: This priority does not guarantee progress in disk collection. It runs disk collection as a very low-level background activity, using spare CPU cycles only. If there are no spare cycles, disk collection waits indefinitely (backs off) until cycles are available or until its priority is increased.
- Priority of 0: This priority has backoff; it has a small impact on system performance and user response time. This priority is the same as that of jobs running in the background.
- Priority of 1: This priority has no backoff; running at this priority will preempt some background jobs.
- Priority of 2: This priority has no backoff; it preempts background jobs.
- Priority of 3: This priority has no backoff; it runs on par with most foreground jobs. This priority has a big impact on system performance, because it shares the same priority as most commands.
- Priority of 4: This priority has no backoff; it preempts most foreground jobs. Editing is still possible, but commands run very slowly.
- Priority of 5: This priority has no backoff; it gives disk collection higher priority than user jobs. Editing is very slow. Command initiation can take up to one minute.
- Priority of 6: This priority has no backoff; it preempts virtually all activity, except that from the operator's console.

MANAGING DISK SPACE

This section covers "how to" commands you can use to manage disk space so that users do not experience a slowdown in system performance or an interruption of their work. These techniques include:

- Displaying the space left on disks
- Changing the priority of an in-progress disk collection
- Checking the current disk-collection priority
- Initiating disk collection explicitly
- Changing disk-collection thresholds
- Determining the current state of disk collection

These operations are covered in separate subsections below.

Displaying the Space Left on Disks

To display the total capacity of each disk and the current amount of free space, you can use the following command from a command window or from the operator's console command interpreter:

```
Operator.Disk_Space;
```

The system produces a display similar to the following:

```
Volume    Capacity   Available    Used     % Free
======    ========   =========   ======    ======
   1        507540      307953   199587      60
   2        538560      292623   245937      54
   3        539946      279663   260283      51
Total      1586046      880239   705807      55
```

The `Volume` column indicates the disk drive. `Capacity` and `Available` show the total space and the free space in terms of pages. `Used` shows the amount of space used on the disk. `%Free` shows the percentage of disk space that is still free. The `Total` line indicates the totals of all disks in each of the fields described above.

Executing the Show_Volume_Summary command from the `Kernel:` prompt results in a more detailed display, as follows:

```
Vol     Total       Unused       Rate
Num   Capacity    Capacity    Blks/Min
-------------------------------------
1       369120      159648        0
2       361680      142940        0
3       391680      156255        0
4       401280      170759        0

low space thresholds for volume 1:
    START_COLLECTION threshold at 15 (waiters exist)
    RAISE_PRIORITY threshold at 14% (waiters exist)
    STOP_JOBS threshold at 12% (waiters exist)
    SUSPEND_SYSTEM threshold at 10% (waiters exist)
    next trigger at 92280 blocks
low space thresholds for volume 2:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 97920 blocks
low space thresholds for volume 3:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
```

```
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 97920 blocks
low space thresholds for volume 4:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 100320 blocks
```

`Vol Num` indicates the disk drive. `Total Capacity` and `Unused Capacity` show total space and free space in terms of pages. `Rate Blks/Min` shows how much space has been used per minute since the last time you executed this command.

The next section of the display indicates the thresholds for each of the volumes. The field at the end of each threshold line (`waiters exist` or `no waiters exist`) indicates whether thresholds have initiated disk collection. The field `waiters exist` indicates that disk collection has not been initiated by a threshold, whereas `no waiters exist` indicates that disk collection has been initiated by a threshold. The last line of the thresholds section of the display (`next trigger at` *n* `blocks`) indicates how much free space (*n* blocks) will be left on the volume when the next threshold is triggered.

Changing the Priority of In-Progress Disk Collection

The job priority for disk collection is represented as an integer value from –1 (the lowest priority) to 6 (the highest priority). These priorities are described in the "Disk-Collection Priorities" subsection, above.

If the job priority of an in-progress disk collection is too high while other jobs are running, users may experience diminished response from the Environment. However, if the job priority of an in-progress disk collection is too low while other jobs are running, disk collection may be too slow to be effective (because obsolete data may be generated faster than it can be collected). Note that if disk collection is not in progress, the procedure described below will have no effect on the disk-collection priority. Also note that you cannot set collection priority lower than the current policy and that this command does not permanently change the priority.

To change an in-progress disk collection that is producing a diminished response from the Environment, execute the !Tools.Disk_Daemon.Set_Current_Priority procedure. For example, to change an in-progress disk-collection priority to 0 on volume 3, enter the following command in a command window or from the operator's console command interpreter:

```
Disk_Daemon.Set_Current_Priority (3,0);
```

Checking the Current Disk-Collection Priority

You may want to check the current collection priority for a given volume before changing the value. For example, to display the current priority of volume 3, enter the following command in a command window:

```
Io.Echo (Disk_Daemon.Current_Priority (3));
```

Initiating Disk Collection Explicitly

You can run disk collection at any time, independently of the Daily client and the automatic threshold mechanism:

- Use the !Commands.Daemon.Collect command to start disk collection on an individual disk at a priority of –1. !Commands.Daemon.Collect also allows you to specify a temporary collection priority—that is, a priority that affects only the disk collection you are about to start.
- Use !Commands.Daemon.Run to run the Disk client on all disks. Disk collection will run at –1 (see the "Disk-Collection Priorities" subsection, above).

For example, to start disk collection on all volumes at priority 6, enter the following command in a command window or from the operator's console command interpreter (remember that 0 indicates all volumes):

```
Daemon.Collect (0, 6);
```

To start disk collection on volume 1 at priority 2, enter the following command in a command window or from the operator's console:

```
Daemon.Collect (1, 2);
```

To start disk collection on all volumes at the –1 priority, enter the following command:

```
Daemon.Run ("Disk");
```

Changing Disk-Collection Thresholds

You can temporarily change the thresholds on any or all disks from the Environment. For example, suppose the current values are:

Start_Collection 25%
Raise_Priority 15%
Stop_Jobs 10%
Suspend_System 8%

To lower the Start_Collection threshold on disk volume 3 to 22%, enter the following command in a command window or from the operator's console command interpreter:

```
Disk_Daemon.Set_Threshold (3, Disk.Daemon.Start_Collection, 22);
```

You must always keep the disk-collection thresholds in descending order from the Start_Collection threshold down to the Suspend_System threshold. For example, suppose the threshold values are ordered as shown above, and you want to lower the Stop_Jobs threshold on volume 3 to 7%. You must first lower the Suspend_System threshold to a value less than 7%:

```
Disk_Daemon.Set_Threshold (3, Disk.Daemon.Suspend_System, 6);
```

Then enter:

```
Disk_Daemon.Set_Threshold (3, Disk.Daemon.Stop_Jobs, 7);
```

If you raise or lower disk-collection thresholds out of order, no error messages will be displayed; however, the thresholds will not be changed from their current values.

Determining the Current State of Disk Collection

To determine the current state of disk collection—such as whether disk collection is running, how it was initiated, whether it is making progress, and which phase it is in—follow the steps described below.

From the EEDB: Prompt

Note: *The steps in this section require that you execute commands from the* `EEDB:` *prompt and the EEDB* `Kernel:` *prompt—not from the R1000* `Kernel:` *prompt. See Chapter 2 for more information.*

1. Get to the console `EEDB:` prompt.
2. From the `EEDB:` prompt, enter the command:

   ```
   kernel
   ```

   The EEDB `Kernel:` prompt appears.
3. Enter the command:

   ```
   show_gc_state
   ```

   A typical console display is:

   ```
   ====>> Elaborator Database <<====
   Kernel: show_gc_state
   DISK daemon has been started on volume 1
     DISK daemon started at 8/26/92 12:48
     Current phase is TAKING_SNAPSHOT
     Current priority is 0 (13)
     Disk daemon allowed to kill an in-progess backup
   ```

   See Table 5-3 for a summary of the messages that may appear on the console.
4. Enter `quit` from the EEDB `Kernel:` prompt to return to the `EEDB:` prompt.

Table 5-3 Disk-Collection Messages Displayed by the Show_Gc_State Command

Message	Phase	Condition
DISK daemon not running		The disk client is not running.
DISK daemon has been started on volume *n*		Where *n* is the number of the volume on which disk collection is running.
DISK daemon started at *mmddyy hh:mm*		Date and time the disk client began.
Current phase is *phase name*		Where *phase name* is the name of the phase in which the Disk client is currently running, such as Traversing-_Virtual_Memory.
Current priority *x((y))*		Where *x* is the current priority at which disk collection is running (–1 is the lowest priority and 6 is the highest priority). Note: *y* is the *real machine priority* and ranges from the lowest priority of 15 to the highest priority of 0. The real machine priority is reserved for Rational.

Table 5-3 Disk-Collection Messages Displayed by the Show_Gc_State Command (continued)

Message	Phase	Condition
z segments have been deleted, so far	Deleting_Segments	Where *z* is the number of segments that the Disk client has deleted.
Collection has backed off a total of *hh:mm*	Traversing_Virtual_Memory	Where *hh:mm* is the number of hours and minutes the disk collector has backed off.
Nonstandard backoff parameters are in effect	Traversing_Virtual_Memory	The Disk client has backed off and nonstandard backoff parameters are in effect.
Backoff when withheld load >= r or run load >= s After backoff, resume when withheld load >= t and run load >= u	Traversing_Virtual_Memory	The Disk client has backed off and standard backoff parameters are in effect, where *r*, *s*, *t*, and *u* are withheld/run loads. For more information on withheld/run loads, see package !Commands.Scheduler in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.
a segments have been visited, so far *b* blocks have been visited, so far	Traversing_Virtual_Memory	Where *a* and *b* are numbers of segments and blocks, respectively, that have been visited.
Footprinting is enabled		Footprinting is a diagnostic tool for use by Rational support personnel.
DISK daemon allowed to kill an in-progress backup		See the "System-Backup Schedule" subsection in Chapter 7 for a description of the !Tools.Disk_Daemon.Set_Backup_Killing command.
DISK daemon not allowed to kill an in-progress backup		See the "System-Backup Schedule" subsection in Chapter 7 for a description of the !Tools.Disk_Daemon.Set_Backup_Killing command.
Problem => DISK daemon is dead		The Disk client can no longer be run.

From a Command Window or from the Operator's Console

To determine whether disk collection is in progress, execute:

```
Daemon.Status ("Disk")
```

If disk collection is in progress, a display such as the following appears:

```
Client     Next Time        Previous Time    Interval    Size     Post     Pre
======   ==============   ==============   ========   ======   ======   ======
Disk     08/09/92 04:00   08/08/92  06:08   01/00:00   745770   654461   923935
Vol 1*   in progress      08/08/92  04:53              218340   167547   310214
Vol 2*                    08/08/92  05:33              289956   266614   341871
Vol 3*                    08/08/92  06:08              237474   220300   271850
```

Pertinent messages in the above display are:

- `in progress`: Indicates that disk collection is occurring on volume 1.
- Asterisks (`*`): Indicates that disk collection is scheduled for volumes 1, 2, and 3 but has not yet completed.
- `Previous Time`: Indicates when the disk daemon last ran. The `Previous Time` field is updated only when disk collection is initiated by the Daily client.

When disk collection is in progress, the Daemon.Status ("Disk") command also produces the Show_Gc_State messages described in the previous section, "From the EEDB: Prompt." These messages appear at the end of the display.

RECLAIMING DISK SPACE

The Disk client automatically reclaims disk space, but it does not remove all redundant data from the disks. Over time, this data can substantially reduce disk capacity and degrade system performance.

To reclaim disk space:

- *Reboot the R1000 weekly:* Regularly rebooting the system reclaims disk space. Certain types of disk collection take place only during system boot operations.
- *Locate space consumers:* Use !Commands.Library.Space to produce a summary by world of disk space. Note that this command requires a minimum of several hours to complete. For worlds that consume large amounts of disk space, consider deleting or archiving certain data.
- *Distribute the storage:* If a volume frequently requires disk collection, consider moving some of the worlds on that volume to other volumes.
- *Make archives:* Archive old CMVC views onto tape and then delete them from disk using the !Commands.Cmvc.Destroy_Views command.
- *Limit the proliferation of temporary and log files:* Delete unneeded files in !Machine.Temporary and !Machine.Error_Logs. ***Note:*** *Retain the files in !Machine.Error_Logs, however, for at least 90 days.*
- *Reduce retention counts:* Set retention counts to 0 on noncritical objects so that each has only one version. Expunge the objects to delete obsolete versions.
- *Expunge deleted objects:* Use Library.Expunge to remove deleted objects from libraries.
- *Compact frequently used libraries:* Use !Commands.Library.Compact_Library to compact the amount of space consumed by frequently modified libraries containing user-created data.

  Caution: *Do* ***not*** *compact the following libraries because they are cached by the system and compacting them causes the listed operations to fail. (This problem is resolved by rebooting. Note that !Machine.Temporary and !Machine.Queues?? are also cached, but the system reenters them into the cache if they are not found.)*

  - !Machine.Users: Logins fail because passwords are destroyed.
  - !Machine.Groups: The group-maintenance commands in package Operator fail with a bad-context (name resolution) problem.
  - !Users: The Login procedure fails to find users' sessions and cannot create replacement sessions, because the system cannot find the !Users library.

PREVENTING RUNAWAY JOBS

A runaway job is a job that uses disk space at a dangerous rate. It is important to prevent runaway jobs by making users aware of page limits and how to control them, as described below.

The Page-Limit Mechanism

The page-limit mechanism is affected by three values:

- The count of pages in memory for the job
- The count of disk blocks consumed by a job in its runtime representation (this does not include the pages generated by Environment operations—for example, creating files)
- The limit for the job

If the number of memory pages or disk blocks consumed by a job exceeds the limit, the job raises Storage_Error exceptions in random locations.

Controls for Page Limits

You can control page limits in four ways:

- Pragma Page_Limit (*x*), where *x* is the number of pages
- Library switch R1000_Cg.Page_Limit
- Session switch Session.Default_Job_Page_Limit
- Procedure !Tools.System_Utilities.Set_Page_Limit

When a library unit is coded, the page limit associated with the unit becomes the maximum of:

- The pragma in the unit, if there is one
- The library-switch value

When a main program is coded, the limit associated with the main program becomes the maximum of:

- The limits associated with the units in the main program's load closure
- The pragma in the unit, if there is one
- The library-switch value

Code is generated so that the main program, when it is called, makes a system call to set its job limit to the maximum of:

- The job's current limit
- The limit associated with the main procedure

In this context, think of a command as an implicit main program, with a library-switch value of 0. Also note that Program.Run, Program.Run_Job, and the operator's console command interpreter behave as commands.

A program can also make calls to !Tools.System_Utilities.Set_Page_Limit to set the limit explicitly. Note that such a call sets the job's page limit to the specified value.

When a job is started, its page limit is initially set to its session-switch value (the default value for the switch is 8,000). When the job executes the program, the program makes calls to further modify the limit, as indicated above.

Note that in a scenario involving multiple main programs, the resulting job page limit depends on which main programs are actually called.

IDENTIFYING AND STOPPING A RUNAWAY JOB

This section describes procedures you can use to determine whether someone has created a runaway job and, if so, what you can do to stop such a job before the system reaches disk-collection thresholds.

Is Disk Space Being Consumed Quickly?

To determine whether a large amount of disk space is being consumed quickly:

1. At the EEDB: prompt, type:

   `kernel`

   to get the EEDB Kernel: prompt.

2. At the Kernel: prompt, enter the command:

   `Show_Volume_Summary`

   A display similar to the following appears:

```
Vol     Total       Unused       Rate
Num   Capacity    Capacity    Blks/Min
-----------------------------------
1       369120      159648       0
2       361680      142940       0
3       391680      156255       0
4       401280      170759       0

low space thresholds for volume 1:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 14% (waiters exist)
    STOP_JOBS threshold at 12% (waiters exist)
    SUSPEND_SYSTEM threshold at 10% (waiters exist)
    next trigger at 92280 blocks
low space thresholds for volume 2:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 97920 blocks
low space thresholds for volume 3:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 97920 blocks
low space thresholds for volume 4:
    START_COLLECTION threshold at 15% (waiters exist)
    RAISE_PRIORITY threshold at 10% (waiters exist)
    STOP_JOBS threshold at 7% (waiters exist)
    SUSPEND_SYSTEM threshold at 5% (waiters exist)
    next trigger at 100320 blocks
```

   `Vol Num` indicates the disk drive. `Total Capacity` and `Unused Capacity` show the total space and the free space in terms of pages. `Rate Blks/Min` shows how much space has been used per minute since the last time you executed this command. A high number in this field, about 1,000 or higher, indicates that disk space is being consumed rapidly. A runaway job possibly exists that must be

stopped before it consumes enough space on that disk to reach the Suspend_System priority.

You should execute the Show_Volume_Summary command several times and watch the `Rate Blks/Min` field. If disk consumption is not high (not greater than about 1,000), there are no runaway jobs.

Is One Job Consuming All of the Disk Space?

If disk space is being used rapidly, the next step is to determine whether one job is using all of the disk space and, if so, which job it is. To determine which job is consuming all of the disk space, you use the Jobs command from the EEDB `Kernel:` prompt.

Note: *The Jobs command should* ***not*** *be entered directly from the R1000* `Kernel:` *prompt—use the EEDB* `Kernel:` *prompt instead. (See Chapter 1 for instructions on accessing and using the EEDB* `Kernel:` *prompt.)*

To determine which job is consuming all of the disk space:

1. At the `EEDB:` prompt, type:

   `kernel`

   to get the EEDB `Kernel:` prompt.
2. At the EEDB `Kernel:` prompt, enter the Jobs command:

   `Kernel: Jobs`

   The following prompt appears:

   `[Threshold [1]:]`
3. Press [Return].

   A display similar to the following appears:

```
Kernel: jobs
Threshold [1]:
 Job Pri Stat CPU%  ModCt Cache   Disk  PgLim DskWts D/S JSegSz WsSiz WsLim
---- --- ---- ---- ------ ----- ------ ------ ------ --- ------ ----- -----
   4   0 R,AT    2   5014  8306  17780  65536 147001   0   4007 10930 11000
   5   0 R,AT    0     11    62     31  65536   1108   0    115   114   200
 169   6 I,CE    0     48   216    220  16000   1434   0     64   116   150
 173   0 I,DT    4      8    18     21   8000   1070   0   5804   131   150
 177   0 I,SV    0      4     6      8   8000     15   0     36    29    75
 181   6 I,CE    0      1     8      0  16000      3   0      0     6    10
 185   6 I,CE    0      1     8      0  16000      1   0      0     6    10
 189   6 I,CE    0      1     8      0  16000      0   0      0     6    10
 193   6 I,CE    0      1     8      0  16000      0   0      0     6    10
 197   6 I,CE    0      1     8      0  16000      0   0      0     6    10
 201   6 I,CE    0      1     8      0  16000      3   0      0     6    10
 205   6 I,CE    0      1     8      0  16000      8   0      0     6    10
 209   6 I,CE    0      1     8      0  16000      5   0      0     6    10
 213   6 I,CE    0      1     8      0  16000      4   0      0     6    10
 217   6 I,CE    0      1     8      0  16000      2   0      0     6    10
 221   6 I,CE    0      1     8      0  16000      4   0      0     6    10
 225   6 I,CE    0      1     8      0  16000      6   0      0     6    10
```

```
 227    0 I,SV     0      17     39      21    8000      73    0       0     52     75
 229    6 I,CE     0       1      1       4   16000      84    0       0      1     10
 231    0 I,SV     0      23     23      60    8000     279    0       1     30     75
 232    6 I,OE     0       1      3       0   16000     178    0      44     63     75
 233    6 I,CE     0       1      5       3   16000      47    0       0      3     10
 235    0 I,SV     0       6     20       2    8000      41    0       1     28     75
 236    0 I,SV     0       2      8       0    8000      34    0       1     13     75
 237    6 I,CE     0       1      3       5   16000      43    0       0      2     10
 240    0 I,SV     0       4      2      28   16000     422    0      76      0      5
 243    0 I,SV     0      28     35      48    8000     108    0       0     52     75
 245    0 I,SV     0       5     28      12    8000     520    0     340     65     75
 247    0 I,SV     0       5      7      10    8000      23    0       0     75     75
 248    0 I,SV     0       3      6       2    8000     101    0       1      8     75
 249    0 I,DT     0     394     68    2052    8000    2020    0       5     44     50
 251    0 I,DT     0      80     15     291    8000     747    0     252     39     50
 252    6 I,CE     0       1      1       7   16000     142    0       0      0     10
 253    0 I,DT     0      66     51     138    8000     313    0       2     31     50
 254    6 I,AT     7      21     62       0    8000    1278    4      77   3515   3550
 255    0 I,SV     0       3      4       9    8000      37    0       3     22     75
Kernel: quit
```

The `Job` field gives the job number for each job. The `Cache` field is the number of pages in main memory consumed by the runtime representation. The `Disk` field indicates the number of pages on disk consumed by the runtime representation. The job is potentially as large as the sum of the `Cache` and `Disk` fields.

The `D/S` field indicates the number of disk waits per second. The `JSegSz` field is the number of pages in the job segment. The `WsSiz` field is the working-set size.

First, you should determine if one of the jobs is large. A job can be large for four reasons:

- A job can be large in its runtime representation. To determine if this is the case for a job, look at the `Cache` and `Disk` fields. A job can be as large as the sum of the values in these two fields. A number greater than 5,000 indicates the job has a large runtime representation.
- A job can have a large job-segment size. If the number in the `JSegSz` field is greater than 5,000, the job has a large job-segment size.
- A job can be large because it is creating a number of files. There is no information in the display that specifically indicates this. To determine accurately if a particular job is creating excessive sizes or numbers of files, you need to know enough about the program's implementation to know where it is likely to be creating such files. Then you can look in the library structure to see if the size or number of files is indeed excessive. A large number in the `D/S` field (30 or more) *may* be significant.
- A job can be creating "temp heaps," which are files that do not appear in the library structure. A large number in the `D/S` field (30 or more) may indicate this.

If one job is consuming a lot of disk space, you should check to see if the disk-space consumption is increasing. To do this, execute the Jobs command several times and watch the appropriate fields for the job of interest.

If the job's space consumption is increasing, you may want to stop it, as described in the following subsections.

4. To return to the `EEDB:` prompt, enter:

```
[quit]
```

Identifying the Job Source

If it appears that you have a runaway job, you should determine which user or other operation generated the job and whether the excessive space consumption is intentional.

You can run several commands to determine which user generated the runaway job, as described below. If there is a runaway job, you should try to stop it before it consumes all available space on the disk.

Using the Job_Names Command

Note: *The Job_Names command should* ***not*** *be entered directly from the R1000* `Kernel:` *prompt—use the EEDB* `Kernel:` *prompt instead. (See Chapter 1 for instructions on accessing and using the EEDB* `Kernel:` *prompt.)*

1. If you are not already at the EEDB `Kernel:` prompt, enter the `kernel` command at the `EEDB:` prompt:

   ```
   ====>> Elaborator Database <<====
   EEDB: kernel
   ```

   The following prompt appears:

   ```
   Kernel:
   ```

2. From the EEDB `Kernel:` prompt, execute the Job_Names command:

   ```
   Kernel: Job_Names
   ```

   The following prompt appears:

   ```
   Threshold [1]:
   ```

3. Press [Return].

   A display similar to the following appears:

```
Threshold [1]:
 Job CPU%     Root   Job Seg Acts   Name
---- ---- -------- --------- ----  -------------------------
   4   14        0   2AA3510   51 System
   5    0        0   2CD490F   17 Daemons
 169    0        0   2CF850F    2 [OPERATOR.GERIB Editor]
 173    7    4A8AD   317D111    1 Mail Distribute Server
 177    0    530B1   317C511    0 Print Queue Server
 181    0        0   2AB7510    1 *Login: 249
 185    0        0   2AB7110    1 *Login: 248
 189    0        0   2AB6D10    1 *Login: 247
 193    0        0   2AB6910    1 *Login: 246
 197    0        0   2AB6510    1 *Login: 245
 201    0        0   2AB6110    1 *Login: 244
 205    0        0   2AB5D10    1 *Login: 243
 209    0        0   2AB5910    1 *Login: 242
 213    0        0   2AB5510    1 *Login: 241
 217    0        0   2AB5110    1 *Login: 240
 221    0        0   2AB4D10    1 *Login: 239
 225    0        0   2AB4910    1 *Login: 238
 227    0    184E3   317C111    0 TCP_IP_Test.Server
 229    0        0   2AB4510    1 *Login: 237
 231    0    1C8E7   317A911    1 Mail SMTP Carrier
 232    0        0   3407D0E    0 [OPERATOR.GERIB Command]
 233    0        0   2AB4110    1 *Login: 236
 235    0    20CEB   3179911    1 Bridge Security Server
```

```
 236     0     33CEC    2CE390F     1 Ftp Server
 237     0         0    2AB3D10     1 *Login: 235
 240     0         0    2CE250F     3 Print Spooler
 243     0     294F3    3176511     0 Teamwork_Interface Dtia Server
 245     0     4F4F5    2AB2910     3 Console Command Interpreter
 247     0     2D8F7    3174911     0 Smooth Snapshots
 248     0     4A4F8    3177511     2 Mail LOCAL Carrier
 249     0     538F9    2AB1510     2 Rcf_Compiler Rev1_1_0
 251     0     340FB    3174111     0 PDL Registration (RATIONAL_2167A, PDL_11)
 252     0         0          0     1 *Login: 16
 253     0     57CFD    2CDED0F     1 DTIA Remote_Operations Server Rev11_4
 254    16     A70FE    2D13D0F     4 !USERS.OPERATOR % ANALYZE_CRASHDUMP_TAPE
 255     0     528FF    316F111     1 Archive Server
Kernel:
```

You should be able to analyze these jobs to determine which is the runaway job. For instance, you may notice that a particular job is taking up an exceptional percentage of the CPU.

Using the What.Users Command

Because many runaway jobs are user jobs, you often can use the What.Users command to match a job number obtained using either the Job_Names or the Jobs command with one of the usernames displayed by the What.Users command.

To execute this command, enter `What.Users` from a command window or from the operator's console. Executing this command produces a display similar to the one below (also see the "Displaying Current Login Information" subsection in Chapter 3):

```
User Status on August 20, 1992 at 3:56:05 PM

     User        Line  Job   S      Time              Job Name
==============  ====  ===  ====  =========  ====================================

*SYSTEM             -    4  RUN     4/21:07  System
                         5  RUN     4/21:07  Daemons
                       163  IDLE   14:55:33  Error Log Monitor
                       193  IDLE    4/20:41  Console Command Interpreter
                       203  IDLE    4/20:45  Print Spooler
                       207  IDLE    4/20:45  Mail SMTP Carrier
                       240  IDLE    4/20:45  Mail LOCAL Carrier
                       242  IDLE    4/20:43  Ftp Server
                       248  IDLE    4/20:38  Problem Report Transfer Server

NETWORK_PUBLIC      -  151  IDLE   14:32:06  Archive Server
                       167  IDLE    2:18:15  Print Queue Server
                       183  IDLE    4/20:39  Mail Distribute Server
                       197  IDLE    4/20:42  Bridge Security Server

OPERATOR          233  153  IDLE  20:25.544  [OPERATOR.SRP Editor]
                       218  IDLE  20:22.236  [OPERATOR.SRP Command]

RACS                -  224  IDLE    2/03:21  CI.INTERPRET("!machine.device ..., ..

RH                249  143  IDLE    3:01:10  [RH.R_3 Command]
                       165  IDLE    2/21:06  Rational_Access User Interface
                       171  RUN       1.251  DEVEL_COMBINED_SMG_02_I_WORKING.UNITS
                       238  RUN     3:01:14  [RH.R_3 Editor]
```

Look in the Job column for the job ID of the runaway job and, if found, match it to the username in the far-left column. You might want to contact this user before you stop the job. Note that there are four types of users listed by the What.Jobs command: *System, Network_Public, Operator, and individual users. Stopping each type of job requires a different technique, as described in the next section.

Stopping a Runaway Job

To stop a runaway job, you may have to use increasing levels of commands before the job actually stops. The procedure below describes the things you should do and the order in which you should do them. If you execute one of these commands to stop the job and your command hangs, use [Control][G] to disconnect from the hung command so you can continue to the next level of effort.

Once you have stopped the job, you will need to run the Disk client. If the running job causes thresholds to be reached (thus initiating the Disk client), the obsolete data for the job will not be collected in that run of the Disk client. Obsolete data collection for that job will have to be done using a second invocation of the Disk client after the job has been stopped.

1. Try to disable the job. From the Kernel: prompt, enter:

   ```
   Disable_Job
   ```

   The system responds with the following prompt:

   ```
   VPID [4]
   ```

2. At the prompt, enter the number of the user job you want to disable and press [Return] (see the "Identifying the Job Source" subsection, above, for information on how to obtain this number).

   If you successfully disable the job, you should try to determine what caused the runaway job. A disabled job may have locks on units that other users need to access. Thus, you may need to proceed to the next step and kill the job so other users can continue working.

3. If you cannot disable the job, try to kill the job. From a command window or from the operator's console command interpreter, enter:

   ```
   Job.Kill (The_Job => [job number], The_Session => "");
   ```

   where [*job number*] is the number of the job to be killed, and The_Session parameter is the user session responsible for the job, as follows:

   - If the runaway job is your own, you need only enter the session number corresponding to the job. For example: S_1.
   - If the runaway job belongs to another user, specify that user's full pathname and session number in The_Session parameter:

     ```
     The_Session => "!Users.Someone.S_3"
     ```

   - If you have determined that the runaway job belongs to *System, enter *System for The_Session parameter:

     ```
     The_Session => "*System"
     ```

   - Certain jobs, such as Print_Queue_Server and Archive_Server, belong to Network_Public. These can be killed using !Machine.Network_Public_Session as The_Session parameter. For example:

     ```
     Job.Kill (The_Job     => 191,
               The_Session => "!Machine.Network_Public_Session");
     ```

Note that in all these cases, you should not try to stop a job using the Job.Kill command more than once (otherwise, you will end up queuing multiple job kills).

4. Contact the appropriate Rational support personnel if none of the above procedures works (see Appendix D for information about how to submit a support request).

Once you have stopped the job, either by disabling the job or by killing it, you should determine what happened in order to prevent another runaway job.

(Chapter 5 continues after the following red tab.)

RECOVERING WHEN THE STOP_JOBS THRESHOLD IS REACHED

In most cases, the best way to recover from the Stop_Jobs threshold is to let the system complete disk collection. In other words, you should let the system recover automatically. You should follow the steps described below *only* if your system has already completed disk collection at least once and the disks have not shown an increase of disk capacity. You may want to monitor the in-progress disk collection to see if the free disk space is increasing during the Stop_Jobs procedure (see the "Determining the Current State of Disk Collection" subsection, above). If the available free disk space does not appear to have increased after the Stop_Jobs procedure has completed, then the system may quickly reach the Stop_Jobs threshold again. In this case, you may want to follow the steps described below (see the "Displaying the Space Left on Disks" subsection, above).

Caution: *The following procedure takes more time to complete than does waiting for the Stop_Jobs procedure to complete. Also, the steps described below involve rebooting the machine; all users are logged off the system, which may result in some users losing uncommitted objects.*

Assuming there is at least 10% free space remaining on your machine and you have concluded that the system will quickly reach the Stop_Jobs threshold again, follow the procedure below. This procedure will cost more than an hour of system unavailability; however, the rebooting may permit the reclamation of additional space (for example, temp heaps and job space) and will stop any runaway jobs.

1. Try to run the Snapshot client to save changes made since the last time the Snapshot client ran. To do this, enter the following command from the EEDB: prompt:

   ```
   Snapshot
   ```

   and wait until the snapshot has finished. Give the system about 5 minutes to try to complete the snapshot. Note that you should *never* run the Snapshot command from the `Kernel:` prompt.

2. Take the system down by pressing [Break].

   A menu similar to the following appears:

   ```
   Please enter
       0 => Restart system
       1 => Ignore break key
       2 => Redisplay recent console output
       3 => Enter debugger
       4 => Reset tape scsi

   Enter option:
   ```

3. At the menu prompt, enter `0`.

   The following prompt appears:

   ```
   Do you really want to crash the system [N]?
   ```

4. At the prompt, enter `Y` and press [Return].

   The system responds:

```
=====================================================
Restarting system after operations console BREAK key

Saving state of: board JKLMFQTVI, Special Registers, Trace Rams
[OK]
R1000 Crash Save done - tombstone file R1000_DUMP1 created.

Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

5. Press [Return] to boot the standard configuration.

See the "Description of the Standard Boot Process" section in Chapter 2 for a detailed description of the standard boot procedure.

RECOVERING WHEN THE SUSPEND_SYSTEM THRESHOLD IS REACHED

The following procedure describes operations you should use to recover if your system reaches the Suspend_System threshold.

1. Turn the operator-mode keyswitch to the Interactive position.
2. Get to the EEDB's `Kernel:` prompt.
3. To determine which volume has reached the Suspend_System threshold, execute the command:

   ```
   Kernel: Show_Volume_Summary
   ```

   Write down the Suspend_System threshold for each volume. You need this information later when you reset the system thresholds.
4. Take the system down by pressing [Break].

   A menu similar to the following appears:

   ```
   Please enter
       0 => Restart system
       1 => Ignore break key
       2 => Redisplay recent console output
       3 => Enter debugger
       4 => Reset tape scsi

   Enter option:
   ```

5. At the menu prompt, enter `0`.

   The following prompt appears:

   ```
   Do you really want to crash the system [N]?
   ```

6. At the prompt, enter `Y` and press [Return].

   The system responds:

```
=====================================================
Restarting system after operations console BREAK key

Saving state of: board JKLMFQTVI, Special Registers, Trace Rams
[OK]
R1000 Crash Save done - tombstone file R1000_DUMP1 created.

Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] : 4
```

7. Enter `4` at the menu prompt and press [Return]. 4
8. Wait until the `Kernel:` prompt appears (this should take approximately 2 minutes). After the prompt appears, execute the command:

   `Kernel: Change_Gc_Thresholds`

   to change the Suspend_System threshold.

   The following prompt appears:

   `VOLUME_NUMBER[1]`
9. At the prompt, enter the number of the disk volume that has reached the Suspend_System threshold.

   The following prompt appears:

   `THRESHOLD[START_COLLECTION]`
10. At the prompt, enter:

    Suspend_System + Stop_Jobs

    The following prompt appears:

    `REMAINING_CAPACITY (%) [10]:`
11. At the prompt, enter `0` for 0%.

    Repeat steps 8 through 11 for each volume that reached the Suspend_System threshold.
12. At the `Kernel:` prompt, execute the command:

    `Kernel: Enable_Priv_Cmds`

    The following message appears:

    ```
    You are enabling a set of commands which must be used
    with extreme care.  They should be used only by
    knowledgeable support personnel.  These commands can
    easily crash/hang the machine; some can completely trash
    the state of the machine such that you must recover the
    machine from backup tapes.
    Proceed [FALSE]:
    ```
13. Answer the prompt with `True`.

    A password prompt appears.

14. Enter the password:

    ```
    Password: secret
    ```

    The `*Kernel:` prompt appears.

15. At the `*Kernel:` prompt, execute the command:

    ```
    *Kernel: Defaults
    ```

    The system resumes booting to the EEDB. In about 45 minutes, the console displays the `EEDB:` prompt.

16. To elaborate to the DDC configuration, execute the command:

    ```
    EEDB: E DDC
    ```

17. After some subsystems elaborate, disk collection starts.

18. Get to the EEDB's `Kernel:` prompt.

19. From the EEDB's `Kernel:` prompt, execute the command:

    ```
    Kernel: Show_Gc_State
    ```

    periodically until you see the message:

    ```
    Disk daemon is not running
    ```

    indicating that disk collection is finished. (Disk collection takes about 40 minutes per volume.)

20. ***Note:*** *You must wait until disk collection finishes before proceeding with this step.* For each disk that was changed in steps 8 through 11, reset the Suspend_System threshold to the original value that you recorded from step 3.

21. Quit the `*Kernel:` prompt:

    ```
    *Kernel: quit
    ```

    The `EEDB:` prompt should reappear.

22. At the `EEDB:` prompt, enter the command:

    ```
    EEDB: E $
    ```

    This command elaborates the standard configuration and resumes the Environment boot process. This step should take 10 to 15 minutes.

    Note: *If this step fails, you may have to crash and reboot the system with the* [Break] *key (see Chapter 2).*

6

Tape Information and Operations

This chapter describes tape formats and tape identification and explains how to handle tape-mount requests. For a description of the 9-track tape drive and the 8-mm tape-cartridge drive, including information about how to mount a tape reel or tape cartridge, see Chapter 15.

As system manager or operator, you may have to answer tape-mount requests executed by other users, both because you must answer these prompts from the operator console and because you might be the only person who has access to the computer room containing the R1000.

This chapter contains sections describing:

- General information about tape formats
- Handling the tape-mount request for writing data to tape
- Handling the tape-mount request for reading data from tape
- Dealing with problems encountered while reading or writing tapes

You respond to tape-mount requests using a generic sequence of steps. Note that writing to tapes and reading from tapes are handled differently. Because of these differences, they are covered in separate sections.

For more details about the specific tape operations that are part of system backups and archiving, see Chapter 7, "Saving and Restoring System Data."

INTRODUCTION TO TAPE FORMATS

ANSI-labeled tapes contain a number of 80-character records, or *labels*, that identify the tape and its contents in conformance with the American National Standards Institute (ANSI) standards for magnetic tape labels. (These standards are defined in "American National Standard Magnetic Tape Labels and File Structure for Information Interchange," ANSI X3.27-1978.) Most Environment tools that read or write tapes use this ANSI standard.

Tapes that are not formatted with ANSI labels are referred to as *unlabeled tapes*. Unlabeled tapes are nonblank tapes with any non-ANSI format, such as a special-purpose format developed by your installation for use on a computer system that cannot handle ANSI formatting. A UNIX *tar* tape is an example of an unlabeled tape. Unlabeled tapes can be read/written by applications using !Io.Tape_Specific.

ANSI-LABELED TAPES

The basic Environment tape commands read and write labeled tapes in two ANSI-based formats:

- Straight ANSI: Typically supported by other computer systems. Its use in the Environment is primarily for reading ANSI-labeled tapes from other systems.
- Chained ANSI: Supported by the Environment and compatible with other computer systems that conform to ANSI-standard labeling. Chained ANSI is used to write tapes on the R1000 and to read those tapes on the R1000 and on other computer systems that conform to ANSI-standard labeling.

Although tapes can be written on the R1000 in straight-ANSI format, chained ANSI is recommended because it is currently the only format that can accommodate multivolume file sets (that is, data that spans more than one tape). Tapes formatted in chained ANSI can be read by any system that can read straight ANSI.

ANSI-labeled tapes use only certain kinds of record formats; these record formats are assigned by default when the standard tape drive is initialized.

Record, straight-ANSI, and chained-ANSI formats are described below.

Record Formats

The three kinds of record format that apply to labeled tapes are:

- Fixed length

  Fixed-length records are all the same length. The block length is also the same for all tape blocks. No indication of the record length is required within the records because the file-header labels define the record length.

  A fixed-length record can be a complete tape block, or several records can be grouped together in a single tape block; a fixed-length record cannot span multiple physical tape blocks.

  When fixed-length records are blocked, it is not necessary that the entire block be filled with data. If a block is not filled, it is padded with caret (^) characters. This means that *you cannot have a fixed-length record containing only carets*; this record will not be interpreted as data.

  When fixed-length records are blocked, I/O efficiency is increased, because many records can be accessed with one I/O request. It is recommended that a block's size be an integral multiple of the record size.

- Variable length

  Variable-length format is used to accommodate files that have records that are not a uniform length. The size of the record is indicated by a record control word (RCW). The RCW consists of four bytes at the beginning of each record.

  A variable-length record can be a complete tape block, or several records can be grouped in a single tape block. A variable-length record cannot span multiple physical tape blocks.

  When variable-length records are blocked, it is not necessary that the entire block be filled with data. If a block is not filled, it is padded with caret (^) characters.

- Spanned

  Spanned format is used to accommodate records that may be larger than the maximum physical block size. A spanned record can occupy more than one physical tape block and it can occupy tape blocks that are on different volumes.

In spanned records, the logical record is represented on tape as a set of variable-length segments that do not span a physical tape block. The first five bytes of each segment are the segment control word (SCW). The SCW consists of a span descriptor byte and four RCW bytes. The four RCW bytes of the SCW indicate the segment length, including the length of the SCW. The segments of a record are written in consecutive order until the entire record is processed.

Note that there is no explicit control word that specifies the total record length. Because records may span multiple volumes and there is no limit to the number of segments in a record, the record length of a spanned record is not bounded.

Straight-ANSI Format

Straight-ANSI format is intended for reading tapes that were written on other computer systems. Straight-ANSI format includes the ANSI labels that support the following facilities:

- Single-volume file sets (that is, one or more files recorded contiguously on a single tape, or *volume*)
- Fixed, variable, and spanned records

Currently, straight-ANSI format does not fully support multivolume file sets (that is, one or more files that span more than one tape).

More specifically, the basic commands in package Tape cannot read multivolume file sets in straight ANSI. (Alternatives for working around this restriction are given in "Reading Multivolume File Sets from Other Systems," below.) Furthermore, multivolume file sets cannot be written in straight ANSI; chained ANSI must be used instead. Note, however, that the restriction on writing poses no real problems, because chained-ANSI tapes can be read on any computer system that supports ANSI-standard labeling.

Finally, Rational's implementation of straight-ANSI format deviates from ANSI-standard labeling in that access control is not supported. The label field that controls accessibility is not interpreted, so there is no way to prevent a tape from being read on the R1000.

Chained-ANSI Format

Chained ANSI is specifically intended to accommodate multivolume file sets. As the name implies, chained ANSI links multiple tapes into a chain. A label at the end of a tape identifies the *next* tape in the chain, and a label at the beginning of a tape identifies the *previous* tape in the chain. The user can read or write information as if using a single tape of infinite length; the user does not have to handle the breaks between volumes (other than to load tapes).

Multivolume file sets written on the R1000 must use chained-ANSI format instead of straight ANSI. However, chained ANSI is not restricted to multivolume file sets; single-volume file sets also can be written in chained ANSI.

In addition to this multivolume facility, chained ANSI supports all straight-ANSI facilities, plus several others. The following list summarizes chained-ANSI facilities:

- Single-volume and multivolume file sets (one or more files recorded contiguously over one or more tapes)
- Fixed, variable, and spanned records
- Filenames up to 532 characters long in addition to the 17-character file identifiers (which are supported by straight ANSI)
- Volume-set names up to 76 characters long that give common identification to each tape in a multivolume file set

Like straight ANSI, Rational's implementation of chained ANSI deviates from ANSI-standard labeling in that access control is not supported, because the label field that controls accessibility is not interpreted.

When to Use Straight and Chained ANSI

Most Environment tape operations use chained ANSI by default. For instance, tape operations that are part of system backup and archiving automatically use only chained-ANSI format. However, the !Commands.Tape.Read and Tape.Write commands allow either straight or chained ANSI to be specified. The following guidelines summarize when to use each kind of format:

- Straight-ANSI format is intended primarily for reading ANSI-labeled tapes from other computer systems. (Currently, basic commands read only single-volume file sets. See "Reading Multivolume File Sets from Other Systems," below.)
- Chained-ANSI format is required for writing data that spans more than one tape and is optional for writing data that fits within a single tape. Note that there is no real need to write tapes in straight ANSI, however, because other systems following the ANSI standard should be able to read chained ANSI by discarding the extra labels.

Reading Multivolume File Sets from Other Systems

If you have a multivolume file set from another computer system and you need to read it on the R1000, you must choose one of the following alternatives:

- Develop Environment tools for reading multivolume file sets using resources from package !Tools.Tape_Tools.
- Break the information on the other system into smaller chunks and rewrite it as a series of single-volume tapes.
- Transfer the data over the network.
- Rewrite the information onto a single larger tape.

Transferring Information between Systems

Many installations use tapes to transfer information between an R1000 and another computer, such as a VAX™, whose operating system supports ANSI tape labeling. Table 6-1 summarizes the tape formats to use for transferring information from one system to the other. The table includes the formats for both single-volume and multivolume file sets.

Table 6-1 Summary of Tape Formats

Direction of Transfer	Single-Volume File Sets	Multivolume File Sets
From another computer to an R1000	Straight ANSI	*
From an R1000 to another computer	Straight or chained ANSI	Chained ANSI only
From R1000 to R1000	Straight or chained ANSI	Chained ANSI only

* See the list of alternatives under "Reading Multivolume File Sets from Other Systems."

For more information about transferring information between systems, see the "Hints for Transferring Information between Systems" section in Chapter 7.

TAPE IDENTIFICATION

Each ANSI-labeled tape is identified by a six-character *volume identifier*, which is written on the tape as a field within a designated label. Volume identifiers can be arbitrary combinations of numbers and letters, or they can encode information such as the date, department, and the like.

Multiple volumes linked together by the chained-ANSI format are further identified by a single *volume-set name*, which is written in a 76-character field on each tape in the set. Whereas the volume identifier serves to identify each tape individually, the volume-set name is typically descriptive and identifies the whole set of tapes. Volume-set names are specified by the user when opening or creating the tape volume.

Unlabeled tapes, as the name implies, are not formatted with ANSI labels. No volume identifier or volume-set name exists on an unlabeled tape.

It is recommended that you affix an identifying adhesive label onto each tape.

How Volume Identifiers Are Assigned

Volume identifiers are assigned only when writing a tape. Installations differ in how and when a volume identifier is assigned to a tape. For example, consider these three installations:

- Installation A follows strict ANSI policy by assigning a unique volume identifier to each tape at the time the tape is initialized. The tape retains that volume identifier no matter how many times the tape is overwritten. Installation A uses an automated tape library system.
- Installation B encodes the current date in each volume identifier, so a new volume identifier needs to be assigned each time a tape is overwritten.
- Installation C follows an informal policy, allowing individual users to assign volume identifiers to their own tapes or allowing the system to generate and assign volume identifiers.

The R1000 accommodates the needs of various installations by providing several strategies for assigning volume identifiers. An installation can use any of the following strategies in any combination:

- Users can assign volume identifiers to their own tapes by specifying the Volume parameter in the !Commands.Tape.Write command. (Assignment by the user takes precedence over all other ways of assigning volume identifiers.)

- The operator can use the !Commands.Tape.Format_Tape command to initialize new tapes, formatting them with ANSI labels that contain specified volume identifiers.
- When responding to the prompts after the mount request for a write operation, the operator can choose among three modes for handling the assignment of the volume identifier. The operator can choose to:
  - Allow the R1000 to generate automatically a unique volume identifier for the tape. The Auto_Generate option specifies this mode.
  - Use the existing volume identifier—that is, preserve the volume identifier that is already on the mounted tape. The Use_Existing_Vol_Ids option specifies this mode.
  - Supply a volume identifier. This choice allows the operator to construct an identifier based on current information (such as the date). The Operator_Supplied option specifies this mode. Volume identifiers assigned using this mode can have the following syntax: The characters must be in the range A–Z or 0–9, each volume identifier can be up to six characters long, and each volume identifier must be separated by commas.

Using Installation A's policy in the example above, an operator would format all new tapes using !Commands.Tape.Format_Tape and then choose to use the existing volume identifier each time the tape is overwritten.

Note: *Users at Installation A should not specify volume identifiers in the !Commands.Tape.Write command, because user-assigned identifiers overwrite existing ones.*

Using Installation B's policy, tapes would not be preformatted. Rather, the operator would supply volume identifiers as tapes were written.

Finally, using Installation C's policy, users would supply volume identifiers with the !Commands.Tape.Write command. If no identifier was supplied, the operator could choose to let the R1000 generate one automatically.

TAPE DENSITY

The 9-track tape drive on the R1000 can read and write tapes in either of two densities:

- Phase encoded (PE), which records 1,600 characters per inch
- Group coded recording (GCR), which records 6,250 characters per inch

The 8-mm tape-cartridge drive reads and writes at a fixed density.

Deciding Which Density to Use

When a 9-track tape is to be read and remote density selection is enabled on the tape drive, the drive automatically switches to the proper density. When a tape is to be written or remote density selection is not enabled on the tape drive, you must manually set the tape drive to the desired density before the write or read operation begins.

The following guidelines will help you choose which density to use:

- Use PE when transporting data to another computer system that can read only PE 1,600 density.
- Use GCR for any other tapes, especially tapes containing large file sets, such as system backups or archive tapes. GCR is recommended over PE because it is more reliable, uses about one-third the amount of tape, and has a higher possible data-transfer rate.

Enabling/Disabling Remote Density Control

You can enable or disable remote density control on the 9-track tape drive by pressing the Density button on the tape control panel. (Pressing this button also sets the tape density used when remote density control is disabled.) Remember the following when deciding whether to enable or disable remote density control:

- When reading a tape, you usually want remote density selection *enabled*, to allow the tape drive to select the proper density automatically. If remote density selection is *disabled*, you must manually select the proper density for reading.

  After mounting a tape, if the Kind of Tape parameter reports that the tape is "unlabeled" or "scratch," but you know there are labels on the tape, remote density selection most likely is *disabled* and the density has been set incorrectly.
- When writing a tape, you will usually want to *disable* remote density selection and manually set the desired writing density. If remote density selection is *enabled*, the density at which the tape was last written will be used regardless of the manual density selection.

CHOOSING AMONG MULTIPLE MOUNT REQUESTS

If several users have entered tape-related commands, multiple mount requests will appear in order on the operator's console, beginning with the oldest request and ending with the most recent.

To process an earlier request first, from the operator's console:

1. Press [Control][Z] to cycle among the requests. Each request is redisplayed in abbreviated form, including only the banner from the request plus the mount question. For example:

   ```
   ====> ANDERSON.S_1 % Tape.Write <====
   Is Tape mounted and ready to read labels?
   ```

   or:

   ```
   ====> ANDERSON.S_1 % Tape.Read <====
   Is Tape mounted and ready to read labels?
   ```
2. If you need more information about a mount request, you can redisplay the entire request by typing `?` and pressing [Return].
3. Keep pressing [Control][Z] until the banner for the desired request is displayed. When you locate the desired request, continue the tape operation by answering the prompts, as described in the following sections.
4. If other prompts appear, press [Control][Z] again to skip them and continue displaying mount requests.

See the "Mount Request for Reading a Tape" section, later in this chapter, when you have located the desired request for a tape-reading operation.

See the following section when you have located the desired request for a tape-writing operation.

MOUNT REQUEST FOR WRITING A TAPE

Most of the procedures that initiate a tape-mount request are:

- Making a full, primary, or secondary backup of the Environment
- Archiving objects onto tape using the Archive.Save command
- Writing ASCII text files or Ada source onto tape using the Tape.Write command
- Writing onto tape with user-generated tools that use package Tape_Tools

Note that your responses to tape-mount requests for writing a tape differ from those for reading a tape. Therefore, the descriptions of how to handle tape-mount requests for reading and writing tapes are in different sections. This section covers tape-mount requests for writing tapes. For more information about handling tape-mount requests for reading tapes, see the "Mount Request for Reading a Tape" section, later in this chapter.

Responding to the Mount Request

Suppose that you or some other user has executed, from a command window or the operator's console, one of the procedures (listed above) for writing onto tape. A mount request will appear on the operator's console. (Note that you do not have to *execute* this command from the operator's console, but you do have to *respond* to the tape request from the operator's console).

A mount request similar to the following is displayed:

```
====> <username>.<session> % <command name> <====
Please Load Tape
  (Kind of Tape     => <kind of tape format>
   Direction        => WRITING,
   Volume Set Name  => <assigned automatically or by user>
   Additional Info  => <omitted unless supplied by user>)

Is Tape mounted and ready to write labels?
```

The main difference between mount requests for tape-writing operations is the kind of information that appears on the initial tape-mount request. The elements and parameters that appear in the mount request for writing a tape are described below.

- Top banner line: Displays the username and session of the user who executed the tape operation and the name of the command that was executed to initiate the tape-mount request. Following are three different examples of a banner line:

```
DRL.S_1 % TAPE.WRITE
OPERATOR.S_1 % SYSTEM_BACKUP.BACKUP
MRB.S_1 % ARCHIVE.SAVE
```

- Kind of Tape: Specifies the kind of tape format used. Most write operations, including making system backups and archives, automatically use chained-ANSI format.

  For more information about the different kinds of tape formats that are possible when writing tapes, see the "Introduction to Tape Formats" section, earlier in this chapter.
- Direction: Specifies whether the tape operation is for writing or reading.
- Volume Set Name: Specifies the volume-set name. This name is assigned automatically or by the user. For example, the backup operation automatically generates a volume-set name that includes the date and time of the backup. For more information about volume IDs, see the "How Volume Identifiers Are Assigned" subsection, earlier in this chapter.
- Additional Info: Allows the user to communicate any other information the user feels may be helpful for tape identification.
- Prompt: The prompt `Is Tape mounted and ready to write labels?` appears at the bottom of the mount request.

Responding to the Mount-Request Prompt

This prompt is your first opportunity to interact with the tape-mount request. All of the information above this prompt was determined when the tape-writing command was initiated; you cannot change any of the parameters in the initial mount request.

You can respond to this question just as you would for a read operation. Four options are:

- If several mount requests are displayed on the operator's console, you can use [Control] [Z] to choose another mount request.
- If the tape drive is already online and a tape is already mounted in it, you can check the availability of the tape drive.
- If you want to refuse the mount request, you can enter no at the prompt.
- If you want to continue the tape operation, you can mount a tape and then enter yes at the prompt.

To choose another mount request, refer to the procedure in the "Choosing among Multiple Mount Requests" section, earlier in this chapter.

To check the availability of a tape drive, refer to the procedure in the "Checking the Drive Availability for Reading" subsection, later in this chapter.

To refuse the mount request, refer to step 2a below and to the procedure in the "Refusing a Mount Request" subsection, later in this chapter.

To mount a tape and continue the tape-write operation, continue in this section.

1. Write-enable a selected 9-track tape reel or 8-mm tape cartridge. Mount it in its tape drive.

   The procedure for mounting a 9-track tape reel in its tape drive is described in the "Mounting a Tape" subsection in Chapter 15. The procedure for mounting

an 8-mm tape cartridge in its tape drive is described in the "Mounting and Dismounting a Tape Cartridge" section in Chapter 15.

2. Respond to the question:

   ```
   Is Tape mounted and ready to write labels?
   ```

   on the operator's console by typing either `yes` or `no` and pressing [Return]. If you want to continue with the tape-writing operation, skip to step 2b below.

   a. If you decide to quit the operation, type `no` and press [Return]. The system responds with the following prompt:

      ```
      Why not? [TRY_AGAIN]
      ```

      You can display a list of possible reasons by entering `?` and pressing [Return]. Part of the list is shown below:

      ```
      Why not? [TRY_AGAIN] ? [Return]
          DESIRED_DRIVE_UNAVAILABLE
          DESIRED_VOLUME_NOT_FOUND
          VOLUME_ACCESS_DENIED
          ...
      Why not? [TRY_AGAIN]
      ```

      Press [Return] to try again (and redisplay the mount request), or type the most appropriate reason and press [Return]. The operation will abort and you will be informed. A corresponding message will also be sent to the Environment terminal of the user who originated the mount request.

   b. If you want to continue with the operation, type `yes` and press [Return].

      The system should respond by displaying information about the tape you mounted. For example, if you are reusing an old backup tape, tape information similar to the following is displayed:

      ```
      ====> OPERATOR.S_1 % SYSTEM_BACKUP.BACKUP <====
      Info on mounted tape is
               (Kind of Tape     => CHAINED_ANSI,
                Writeable        => TRUE,
                Volume Id        => 007701,
                Volume Set Name => BACKUP, 31-AUG-92 21:02:36 3)
      ```

      If your operator's console does not show this display but shows a single-line error message instead, something is incorrect. To determine the cause of the problem, see the discussion of error messages in the "Handling Problems Encountered in Reading/Writing Tapes" section at the end of this chapter.

Continuing with the Tape Mount

1. Respond to the question:

   ```
   OK to overwrite volume? [YES]
   ```

   If you decide not to overwrite the tape you mounted, enter `no` and press [Return].

   The mount request is redisplayed and you can mount another tape. To overwrite the mounted tape, press [Return]. Continue with the next step.

2. Choose a method of assigning a volume identifier to the tape:

   ```
   What should the volume id handling mode be? [<default mode>]
   ```

   Enter one of the following and press [Return]:

- `Auto_Generate`

  Automatically generates a unique volume identifier for this and subsequent tapes in the volume set. This option is the easiest choice for multivolume sets.

- `Use_Existing_Volume_Ids`

  Preserves the volume identifier that is already on the tape you mounted. This option cannot be used for a continuation volume (refer to "Writing a Continuation Volume," below).

- `Operator_Supplied`

  Allows you to enter a list of volume identifiers to be used on this and subsequent tapes in the volume set. With this option, you can construct informative volume identifiers or you can supply existing volume identifiers from preformatted tapes.

Regardless of the choice you make, the system eventually responds by assigning a volume identifier and then displaying it. For example, if you answer the following prompt with `Auto_Generate` **[Return]**:

```
What should the volume id handling mode be?
```

a prompt similar to the following is displayed:

```
Volume id of tape is now: 007802
```

Auto_Generate creates a unique volume identifier.

If you are using the 8-mm tape-cartridge drive, no further operator input is required. Skip to the next subsection, "Removing the Tape."

3. If you are using the 9-track tape drive, a prompt similar to the following will be displayed:

```
Tape will be written in PE_1600CPI.
If not correct, change at drive.
```

This message reminds you to set the correct recording density and density control on the 9-track tape drive if you have not already done so. Disregard the message if you want to write on the current tape using the displayed density (which is the density used when this tape was last written).

If you are not sure that the drive will use the correct density and/or density control, this is your last opportunity to check it. To change the density and/or the density control on the 9-track tape drive, repeatedly press the Density button on the tape control panel until the desired density and density control are displayed on the tape control panel. (For more details, see "Setting Tape Density and Local/Remote Density Control" in Chapter 15.)

Removing the Tape

When the write operation completes or reaches the end of the tape, the tape is automatically rewound onto its supply reel. If the drive is an 8-mm cartridge-tape drive, the tape-drive door opens automatically. Then a prompt similar to the following is displayed:

```
====> ANDERSON.S_1 % Tape.Read <====
Please dismount Tape on Drive: 0
 (Kind of Tape     => CHAINED_ANSI,
  Volume Id        => 005020)
```

Note the line that reads: `Please Dismount Tape on Drive: 0`. This means that the tape operation is complete and you can now remove the 9-track tape reel or 8-mm tape cartridge from the drive.

If more tapes are needed when writing a chained-ANSI volume set, see the next subsection, "Writing a Continuation Volume."

Writing a Continuation Volume

If you are using chained-ANSI tapes and the tape operation requires more than one 9-track tape reel or 8-mm tape cartridge, the volume identifier for the next tape is written at the end of the current tape. You will be prompted if no volume identifier is available for the next tape. For example, if no volume identifier is supplied, the operator's console might display the following message:

```
Continuation volume needed.
What should the volume id handling mode be? [AUTO_GENERATE]
```

1. Specify the volume-identifier mode. Note that you cannot specify the Use_Existing_Volume_Ids option as the volume-ID handling mode. This option preserves the existing volume identifier, and it can be used only for the first of multiple tapes in a volume set.

   After you specify the volume identifier, the current tape is unloaded automatically and the tape removal prompt is displayed, followed by a mount request for the next tape.

2. Remove the current tape and repeat the steps in this section ("Mount Request for Writing a Tape"). Note that one step, assigning a volume ID, is omitted as long as a volume identifier is available for the next tape.

Note: *Never manually remove a tape when executing a tape operation. The tape drive will always automatically unload the tape at the right time. If you must manually unload a tape at any time during a tape operation, something has gone wrong and you should restart the entire operation.*

MOUNT REQUEST FOR READING A TAPE

Most of the procedures that initiate a tape-mount request are:

- Restoring the Environment from backup tape(s)
- Archiving objects or Ada units from tape to the R1000 using the Archive.Restore command
- Reading ASCII text files or Ada source from tape to an R1000 using the Tape.Read command
- Reading from tape to an R1000 with user-generated tools that use package Tape_Tools

Responding to the Mount Request

When a tape-mount request appears on the operator's console, you are responsible for mounting a tape that matches the information appearing in the mount request. Your main responsibility is to identify the correct tape.

Suppose a user has executed a procedure or issued a command that requires a tape to be read. A mount request appears on the operator's console. (Note that you do not have to execute the tape-reading command from the operator's console, but you do have to respond to the tape request from the operator's console.)

A mount request similar to the following is displayed:

```
====> <username>.<session> % <command name> <====
Please Load Tape
  (Kind of Tape     => <kind of tape format>,
   Direction        => READING,
   Volume Id        => <only displayed for labeled tapes>,
   Additional Info => <omitted unless supplied by user>)

Is Tape mounted and ready to read labels?
```

The main difference in mount requests for tape-reading operations is the information that appears on the initial tape-mount prompt. The data that appears in the mount request for reading a tape is described below:

- Top banner line: This line displays the username and session of the person who executed the tape operation and the name of the command that was executed to initiate the tape-mount request. Following are two different examples of a top banner line:

  ```
  JAF.S_1 % TAPE.READ
  SJY.S_1 % ARCHIVE.RESTORE
  ```

- Kind of Tape: This parameter specifies the kind of tape format expected. Note that most write operations, including making system backups and archives, automatically format tapes in chained-ANSI format.
- Direction: This parameter specifies whether the tape operation is for writing or reading.
- Volume Id: This parameter displays the volume identifier of the tape to be read. If the tape cartridge mounted is an unlabeled tape, then no volume identifier has been specified and this field is omitted.

  For more information about volume IDs, see the "How Volume Identifiers Are Assigned" subsection, earlier in this chapter.
- Additional Info: This parameter allows you to communicate any other information that may be helpful for tape identification.
- Prompt: The prompt `Is Tape mounted and ready to read labels?` appears at the bottom of the mount request.

This prompt is the first opportunity you have to interact with the tape-mount request. All of the information above this prompt was determined at the time the tape-reading request was initiated. You cannot change any of the parameters in the initial mount request. Follow the procedure below to mount the correct tape for reading.

Checking the Drive Availability for Reading

When you mount a tape and respond to a mount request, the tape drive is *allocated* to the user who generated the request. Many tape-related commands automatically deallocate the tape drive after they finish. However, some commands allow the drive to remain allocated to the same user between successive tape operations. In this case, the drive is not deallocated until the user requests deallo-

cation, the user logs out with no running background jobs, or the tape drive is taken offline.

If the tape drive is online and a tape is already mounted, the tape drive may still be allocated to another user. You can check whether the tape drive is free or allocated before you handle the current mount request. To do this:

1. Leave the previous tape in the tape drive.
2. At the following prompt on the operator's console, type `yes` and press [Return]:

   ```
   Is Tape mounted and ready to read labels?
   ```

3. If a message similar to the following is displayed:

   ```
   Drive 0 is busy
   ```

   then the tape drive is still allocated to the previous user. If you see this message, refer to the "Handling Problems Encountered in Reading/Writing Tapes" section, at the end of this chapter.

Mounting the Tape to Be Read

After you have determined that the tape drive is deallocated and available:

1. Locate the requested tape cartridge. If you want to prevent accidental writing on this tape, ensure that the reel or tape cartridge is not write-enabled. Mount the reel or tape cartridge on the tape drive.

   The procedure for mounting a 9-track tape reel in its tape drive is described in the "Mounting a Tape" subsection in Chapter 15. The procedure for mounting an 8-mm tape cartridge in its tape drive is described in the "Mounting and Dismounting a Tape Cartridge" section in Chapter 15.

2. Verify that the tape drive is online.

   If the tape is mounted and the tape-drive door is closed, the tape drive should come online within about 30 seconds (a green light comes on).

3. After the following prompt appears on the operator's console, type `yes` and press [Return]:

   ```
   Is Tape mounted and ready to read labels?
   ```

Verifying the Mounted Tape

Once you have mounted a tape and answered `yes` to the mount question, the mounted tape is inspected to verify that it matches the requested tape, and information from the mounted tape should be displayed on the operator's console. (If this information is not displayed, see the "Handling Problems Encountered in Reading/Writing Tapes" section at the end of this chapter.)

Information from the mounted tape is shown in a display similar to the following:

```
====> <username>.<session> % <command name> <====
Info on mounted tape is
  (Kind of Tape     => <kind of tape format>
   Writeable        => FALSE
   Volume Id        => <only displayed for labeled tapes>
   Volume Set Name  => <omitted unless chained ANSI format>)
```

The display includes the applicable fields for the tape cartridge you have mounted. These fields are:

- Kind of Tape: Indicates the format of the mounted tape. You will typically see `CHAINED_ANSI`. However, if you mount a tape from another computer system, you might see:
  - `STRAIGHT_ANSI`

    The tape is a single-volume file set and may have been written on another computer to be read on the R1000.
  - `UNLABELED_TAPE`

    The tape has a non-ANSI format.
  - `SCRATCH`

    The tape is blank or is written in a density that the R1000 tape drive cannot read.
- Writeable: Indicates whether the mounted tape has been write-enabled: True or False. The value of this field is irrelevant for read operations. However, if this field indicates that the tape is write-enabled, users will be able to overwrite the tape (because the tape drive is now allocated).
- Volume Id: Displays the volume identifier of the tape to be read. If the tape cartridge mounted is an unlabeled tape, then no volume identifier has been specified and this field is omitted.
- Volume Set Name: Displays the volume-set name if the mounted tape has chained-ANSI format; otherwise, this field is omitted.

The mounted tape matches the requested tape if the volume identifiers and formats are compatible. More specifically:

- If a volume identifier was specified in the mount request, the volume identifier on the mounted tape must match it exactly. If no volume identifier was specified in the mount request, the mounted tape can have any volume identifier.
- If the format displayed in the mount request is Chained_Ansi, the mounted tape must have a chained-ANSI format.
- If the format displayed in the mount request is Straight_Ansi, the mounted tape can have either straight- or chained-ANSI format.
- If the format displayed in the mount request is Unlabeled_Tape, the mounted tape can have any format. (The mount request displays `UNLABELED_TAPE` in response to the !Commands.Tape.Display_Tape command.)

If the Tape Matches

If the mounted tape matches the requested tape, the following prompt appears after the tape information, asking you to confirm that the read operation should start:

```
OK to read volume? [YES]
```

This is your last opportunity to stop the read operation. For example, even if the mounted tape is compatible with the requested tape (as determined by the above criteria), you may decide that it is not the correct tape, based on other information.

To stop the read operation and redisplay the mount request, type no and press [Return]. From the mount request, you can cancel the operation (see the "Refusing a Mount Request" subsection, later in this chapter) or try another tape.

To start the read operation, press [Return]. The tape is read and, when the operation finishes, the message for dismounting the tape is displayed. To dismount the tape, refer to the "Dismounting the Tape" subsection, later in this chapter.

If the Tape Does Not Match

If the mounted tape does not match the requested tape, an error message is displayed immediately after the tape-information display, followed by the original mount-request display. The read operation cannot continue. For example, if you unintentionally mounted a tape with a non-ANSI format, a display similar to the following would appear:

```
====> ANDERSON.S_1 % Tape.Read <====
Info on mounted tape is
  (Kind of Tape     => UNLABELED_TAPE,
   Writeable        => TRUE)
Request does not match tape.

====> ANDERSON.S_1 % Tape.Read <====
Please Load Tape
  (Kind of Tape     => STRAIGHT_ANSI,
   Direction        => READING,
   Volume Id        => 005020)
Is Tape mounted and ready to read labels?
```

Compare the information displayed about the tape and the mount request. Depending on the problem, you can:

- Mount the correct tape and follow the tape-mounting procedure again.
- Check whether the user supplied incorrect information about the tape. For example, if you mounted the tape with the specified volume identifier but the formats did not match, then the user incorrectly specified the format. In this case, you might refuse the mount request and use the !Commands.Message.Send command to tell the user to reenter the !Commands.Tape.Read command with the correct information.

Dismounting the Tape

When the read operation completes, the tape is automatically rewound onto its supply reel. If the drive is an 8-mm tape-cartridge drive, the tape-drive door opens automatically. Then a prompt similar to the following is displayed:

```
====> ANDERSON.S_1 % Tape.Read <====
Please Dismount Tape on Drive: 0
 (Kind of Tape     => CHAINED_ANSI,
  Volume Id        => 005020)
```

Note the line that reads, `Please Dismount Tape on Drive: 0`. This means that the tape operation is complete and you can now remove the 9-track tape reel or 8-mm tape cartridge from the drive.

Refusing a Mount Request

Sometimes you will not be able to mount a tape because the tape drive is already allocated or you cannot find the requested tape. To refuse a mount request:

1. At the following prompt on the operator's console, type no and press [Return]:

   ```
   Is Tape mounted and ready to read labels?
   ```

   The system responds:

   ```
   Why not? [TRY_AGAIN]
   ```

2. You can display the list of responses recognized by the system by typing ? and pressing [Return]. Note that only part of the list is shown below:

   ```
   Why not? [TRY_AGAIN] ? [Return]
   DESIRED_DRIVE_UNAVAILABLE
   DESIRED_VOLUME_NOT_FOUND
   VOLUME_ACCESS_DENIED
       ...
   Why not? [TRY_AGAIN]
   ```

3. Press [Return] to try again and redisplay the mount request, or type the most appropriate reason and press [Return]. The operation will abort and you will be informed. A corresponding message also will be sent to the Environment terminal of the user who originated the mount request.

HANDLING PROBLEMS ENCOUNTERED IN READING/WRITING TAPES

You may encounter several kinds of problems while reading or writing a tape. Some involve problems with the tape drive and others result from using bad tapes. Each type of error is described in a separate subsection below.

Wrong Tape

If you mount the wrong tape on a tape drive that has already been deallocated, the system responds by trying to continue the current operation with the previous user's tape:

- Information about the mounted tape is displayed.
- If the mounted tape is compatible with the requested tape (for example, no volume identifier was specified in the request, so any tape could match), the following question is displayed:

  ```
  OK to read volume? [YES]
  ```

- Enter no and press [Return]. The current mount request is redisplayed.
- If the mounted tape is not compatible with the mount request (for example, conflicting volume identifiers), then a message indicates that the read operation has failed and the current mount request is redisplayed.

  In any case, dismount the incorrect tape by pressing the Unload button. Mount the correct tape so that you can continue with the tape-reading or (tape-writing) operation.

Drive Is Busy

During a tape-reading or tape-writing procedure, when you answer the following prompt affirmatively:

```
Is Tape mounted and ready to read labels?
```

the operator's console might display a message alerting you that the tape drive is busy. For example:

```
Drive 0 is busy
```

This means that the tape drive is still allocated to another user.

The mount request is then redisplayed. To correct the problem, do one of the following:

- Process the mount request later, after the present user has deallocated the tape drive.
- Leave the drive allocated to the present user and cancel your mount request. (For the procedure, see the "Refusing a Mount Request" subsection, above.)
- Deallocate the tape drive by unloading the present user's tape and then continue with your mount request.

Drive Not Online

During a tape-reading or tape-writing procedure, if you answer the following prompt affirmatively:

```
Is Tape mounted and ready to read labels?
```

before mounting the tape or when the tape drive is not online, the following message appears on the operator's console:

```
Error encountered while reading labels: NOT_ON_LINE
```

The current mount request is then redisplayed. Correct the problem by mounting a tape, closing the tape-drive door, and waiting for the drive to come online. Then proceed with the rest of the tape-mount request.

TROUBLESHOOTING TAPE ERRORS

A number of different errors may occur when trying to read from or write to a tape. These errors may be recoverable (this means that the system corrects them automatically) or unrecoverable (that is, the system cannot correct them and aborts whatever tape operation was in progress). Some problems that halt a tape operation are corrected easily by the operator—such as putting the tape drive online. These are not the kind of tape errors described below, because they do not involve problems with the tape media or the tape drive itself. (For other ideas, see the "Handling Problems Encountered in Reading/Writing Tapes" section, above.)

The three most common causes of tape errors are (in order):

- Dirty heads on the tape drive
- Old or damaged tape
- Malfunctioning tape drive

Dirty Tape Drive

If you notice that many successive tapes, even new ones, are causing tape operations to fail, the drive may need cleaning. In this case, after the tape drive has been cleaned, you will sometimes be able to read a tape that previously failed. See the "Tape-Drive Maintenance" subsection in Chapter 15 for information on the care of the tape drive.

Unreliable Tape

Usually, when you see that a tape operation (such as making a backup) fails near the end of the operation, a bad tape should be suspected.

If a tape operation fails because of a bad tape, you will see the following message on the operator's console:

```
Unexpected_Tape_Error
```

If this error occurs, you will have to restart the operation. However, before doing so, you should clean the drive and replace the tape with a tape you know is good (either a new tape or one that has been verified).

For information that will help you ensure tape reliability, see the "Proper Care of 9-Track Tapes" subsection and/or the "Tape-Cartridge Reliability" subsection in Chapter 15.

Malfunctioning Tape Drive

If a tape drive actually experiences a hardware failure, the drive will send the following message to the operator's console:

```
Unit_Is_Bad
```

If you see this message during any phase of a tape-read or tape-write operation, contact your Rational support representative. For details about how to submit a support request, see Appendix D.

Other Troubleshooting Techniques

It is sometimes possible that a tape cannot be read or written on one tape drive, but it can be read or written on a different tape drive. First, locate a known good tape that has been written on by a known good tape drive. Try to read and write/read this tape on the suspect tape drive. If it reads and writes properly, the original tape is probably faulty. If the good tape does not read or write/read properly, the drive probably has a problem. If you have access to another Rational system, you may want to retry the operation on the other system. The results of this may help you verify whether the problem is with the tape or the drive. If the same tape fails with the same error on two different drives, it is probably bad.

7

Saving and Restoring System Data

This chapter describes different methods the Rational Environment provides for saving system information to protect against accidental loss of data. The two main methods are:

- *System backup:* Saves a bit-by-bit copy of the entire Environment state on one or more tapes. Regular system backups ensure that the entire Environment—but not individual objects—can be restored with minimal loss after a catastrophic system or Environment failure.
- *Archive:* Saves one or more selected objects onto tape. Regular archiving of important objects ensures that recent versions of these individual objects can be recovered after accidental deletion, overwriting, and the like.

System backup and archive operations have different but complementary capabilities. A backup allows you to record and restore the entire Environment—for example, in the event of disk failure. However, you cannot use a backup tape to restore a selected subportion of the Environment, such as a single Ada unit that has been accidentally expunged. In contrast, archiving operations permit you to record and restore specific objects, such as worlds, directories, Ada units, or text files. Archive tapes also can be used to transfer specific portions of the library system from one R1000 to another.

System backups typically are taken by the operator or system manager on a regular basis. In contrast, archiving of individual objects or project worlds can be performed by users, typically on an as-needed basis. However, because restoration from an archive is so flexible, some installations may choose to supplement system backups by regularly archiving important user and project worlds. Note that archiving should not be used *instead* of the recommended system backups because archive tapes are not suitable for the restoration of the entire Environment. Furthermore, archiving takes considerably longer than system backups, so regularly archiving the entire Environment is not practical.

This chapter also describes different ways to safeguard your data once it is saved on tape. See the "Safeguarding System Information" section later in this chapter for more details about verifying data tapes, examining the system backup log, examining the tape labels, and storing tapes off site.

Finally, this chapter also describes useful, but less common, tape operations. See the "Other Tape Operations" section at the end of this chapter for information about:

- Making a DFS backup tape
- Making a SIMS log tape
- Using the Tape.Read and Tape.Write commands
- Transferring information between systems

SYSTEM BACKUPS

A system backup copies a snapshot of the Environment onto a set of ANSI-compatible tapes. The tapes then can be read during the system boot process to restore the Environment after a catastrophic failure.

The Environment supports three kinds of backups:

- *Full:* Saves the complete Environment, including all user data and system information. Full backups are complete and self-sufficient. A full backup must be the first backup taken after the system has been restored from backup tapes.
- *Primary:* Saves any changes since the last full backup. A primary backup cannot be the first backup on a restored system.
- *Secondary:* Saves any changes since the last primary backup. A secondary backup cannot be the first backup on a restored system.

System-Backup Schedule

Rational recommends the backup schedule shown in Table 7-1. (This schedule can be used in conjunction with the schedule described in the "Off-Site Tape Storage" subsection, later in this chapter.)

Table 7-1 Recommended Backup Schedule

Type of Backup	Frequency
Full backups	Once per week
Primary backups	Once per day*
Secondary backups	Seldom, if ever

* Cannot be the first backup on a restored system—always associated with most recent full backup.

The following are recommended in conjunction with system backups:

- Because backups adversely affect system response for user jobs, schedule backups when the system load is expected to be low.
- With a 9-track tape drive, use only the highest-quality 2,400-foot tapes. With an 8-mm tape-cartridge drive, use only the cartridges listed in Table 15-6.
- Schedule system backups so they do not overlap Disk-client operations. (See "The Disk Client" section in Chapter 5.) By default, if disk collection starts during a system backup, the Disk client kills the backup so that disk collection can proceed.

In general, do not disable the Disk client's ability to disable a backup.

Caution: *If backup killing is disabled, it is important that you determine the available disk space on the system before performing a backup (see the subsection on "Displaying the Space Left on Disks" in Chapter 5). If there is less than 10% free space on the system when the backup is made, the backup cannot be restored on the same system. Normally, the system will cross the Start_Collection threshold before*

this can happen. Likewise, if backup killing is enabled on your system, make certain that the backup and the Disk client do not run at the same time, because the Disk client will kill any in-progress backup.

By default the Disk client allows backup killing. You change this by running the following command:

```
Disk_Daemon.Set_Backup_Killing (False)
```

When backup killing is set to False, an in-progress backup prevents disk collection from running. (For more information, see "Automatic Disk-Collection Thresholds" in Chapter 5.) If you set backup killing to False so you can run an uninterrupted backup, remember to run the following command as soon as you finish the backup:

```
Disk_Daemon.Set_Backup_Killing (True)
```

This restores the Disk client's ability to kill an in-progress backup.

To query the current value of backup killing (True or False), run the following command:

```
Io.Echo (Boolean'Image(Disk_Daemon.Disk_Daemon.Backup_Killing_
                                                    Enabled))
```

System-Backup Tapes

A backup records two kinds of information:

- Data, such as the contents of Environment objects
- All other system information, such as information representing the system configuration

  On a 9-track tape drive, this system information is written on a separate short tape that was formerly called a *blue tape*. It is now called a *backup-index tape*.

  On an 8-mm tape-cartridge drive, this system information is appended to the last tape in the backup volume set, in a *backup index*. Note that the information written in the backup index on an 8-mm cartridge is exactly the same as the information written on the separate 9-track backup-index tape.

With a 9-track tape drive, most backups typically require multiple 2,400-foot tapes (plus the separate backup-index tape when making a system backup).

With an 8-mm tape drive, most backups typically require only one tape cartridge to record all of the system data (and the backup index when making a system backup).

The system-backup operation automatically assigns a *volume-set name* to the data tape. This volume-set name includes the date and time of the backup as well as the backup version. For example:

```
BACKUP, 04-AUG-92 21:08:49 3
```

With a 9-track tape drive, the system-backup operation also automatically assigns a volume-set name to the backup-index tape. For example:

```
BACKUP BLUE TAPE, 04-AUG-92 21:08:49 3
```

Making a System Backup

When you make a system backup, you will probably use the operator's console, although you can use any Environment session to initiate the backup procedure. Initiating a backup procedure displays a tape-mount request on the operator's console. After you mount a tape on the tape drive and respond to the prompts on the operator's console, the Environment begins writing data onto the tape. If more than one tape is required, you will be prompted on the operator's console for another tape.

Starting a Full Backup

1. To begin the backup immediately, enter the following command from the operator's console command interpreter or an Environment session:

   ```
   Full_Backup;
   ```

   Or, to delay starting the backup to a specified time, use:

   ```
   Full_Backup("<specified time>");
   ```

   (For example, if you want the backup to begin at 7:00 P.M., enter `19:00` as the specified time.)
2. Continue with "Handling the Backup Tape-Mount Request," below.

Starting a Primary Backup

1. To begin the backup immediately, enter the following command from the operator-console command interpreter or an Environment session:

   ```
   Primary_Backup;
   ```

   Or, to delay starting the backup to a specified time, use:

   ```
   Primary_Backup("<specified time>");
   ```

   (For example, if you want the backup to begin at 7:00 P.M., enter `19:00` as the specified time.)
2. Continue with "Handling the Backup Tape-Mount Request," below.

Starting a Secondary Backup

1. To begin the backup immediatly, enter the following command from the operator's console command interpreter or an Environment session:

   ```
   Secondary_Backup;
   ```

   Or, to delay starting the backup to a specified time, use:

   ```
   Secondary_Backup("<specified time>");
   ```

   (For example, if you want the backup to begin at 7:00 P.M., enter `19:00` as the specified time.)
2. Continue with "Handling the Backup Tape-Mount Request," below.

Suggestions for Building a Customized Backup Procedure

You can use the procedures described below by themselves, or you can call the !Commands.System_Backup.Backup procedure as part of a customized procedure that controls more aspects of the backup process.

One such customized procedure is !Commands.Abbreviations.Do_Backup, which is provided with the Environment. You can modify this procedure or build one of your own.

One advantage of a customized backup procedure is that you can use it to start the backup without the usual delay caused by the snapshot warning interval. That is, because the System_Backup.Backup command takes a snapshot, the backup must wait for the snapshot warning interval to elapse (see "The Snapshot Client" in Chapter 5). To avoid the wait, start the backup with the Do_Backup procedure, which sets the snapshot warning interval to 0, executes System_Backup.Backup, and then resets the interval to 2 minutes. (You can modify this procedure to reset the snapshot warning interval to the value used at your installation.)

To use the Do_Backup customized procedure, enter one of the following (omitting the parameter specifies a full backup):

```
Do_Backup;
Do_Backup (System_Backup.Primary);
Do_Backup (System_Backup.Secondary);
```

Further uses for a customized backup procedure include these:

- You can include the !Commands.Message.Send_All procedure to broadcast messages about the backup to all users. For example, you can inform users when the backup will begin, advise users to commit any objects that they want recorded in the backup, or request users to log out.
- You can include calls to procedures such as Operator.Force_Logoff or Operator-.Disable_Terminal to log out current users or to prevent users from logging in.

The second parameter for the Do_Backup procedure takes an optional date and time for the backup. This second parameter allows you to initiate a system backup and mount a backup tape during normal hours of operation, leaving the backup to proceed during off hours, when it will not interfere with users or system daemons. This feature is useful for making backups that require only a single tape.

The following command takes a backup at 11:00 P.M. the same day:

```
Do_Backup (Starting_At => "23:00");
```

The message announcing the backup is displayed on the operator's console when the actual backup begins. The time specified by the Starting_At parameter is absolute, rather than relative to the time at which the tape is mounted; consequently, the backup begins at the Starting_At time no matter when the tape is mounted (as long as it is mounted before the backup is scheduled to start).

The following command takes a primary backup at 2:00 A.M. on July 10, 1992:

```
Do_Backup (System_Backup.Primary,"92/07/10 2:00:00");
```

The following command takes a secondary backup at 2:00 A.M. on July 11, 1992:

```
Do_Backup (System_Backup.Secondary,"92/07/11 2:00:00");
```

The Backup Progress Message

After you have executed any kind of backup operation, the console from which the backup was started will display progress messages similar to the following:

```
------------------------------------------------------------------
!USERS.DRL % DO_BACKUP                          STARTED 10:49:51 AM
------------------------------------------------------------------

10:50:::[System_Backup.Backup(Variety=>SECONDARY,Wait_Until=>"")].
10:50 --- Requesting the operator to mount the 1st tape.
10:55 --- Backup is Starting.
10:55 --- Starting retained snapshot.
10:56 --- Retained snapshot has completed.
10:56 --- Writing space information for disk volume 1.
11:17 ---  681 blocks written.
11:17 --- Writing block information for disk volume 1.
11:17 ---  51 blocks written.
11:17 --- Writing block data for disk volume 1.
11:21 ---  4810 blocks written.
11:22 --- Writing space information for disk volume 2.
11:41 ---  890 blocks written.
11:41 --- Writing block information for disk volume 2.
11:41 ---  33 blocks written.
11:41 --- Writing block data for disk volume 2.
11:45 ---  3066 blocks written.
11:45 --- Writing space information for disk volume 3.
12:03 ---  668 blocks written.
12:03 --- Writing block information for disk volume 3.
12:04 ---  111 blocks written.
12:04 --- Writing block data for disk volume 3.
12:14 ---  10319 blocks written.
12:14 --- Writing space information for disk volume 4.
12:30 ---  705 blocks written.
12:30 --- Writing block information for disk volume 4.
12:30 ---  60 blocks written.
12:30 --- Writing block data for disk volume 4.
12:35 ---  5657 blocks written.
12:35 --- Requesting the operator to mount the blue tape.
12:36 --- Writing the Backup Index.
12:37 --- Backup Index contains 101 blocks in 6 files.
12:38 +++ Backup has completed.
12:38 +++ Running Snapshot daemon.
```

In the foregoing example, the operation was for a *secondary* backup on the 9-track tape drive. Similar progress messages are displayed for every kind of back-up operation.

If the backup is being done on the 8-mm tape-cartridge drive, the line requesting the backup-index tape (the blue tape) does not appear.

Handling the Backup Tape-Mount Request

Shortly after you start any kind of backup by executing the System_Backup-.Backup procedure or a customized backup procedure, a mount request for the first data tape appears on the operator's console. Handling the mount request is the same for full, primary, or secondary backups. The procedure described below summarizes how to handle the backup tape-mount request. If you encounter errors while making a system backup, refer to the "Handling Problems Encountered in Reading/Writing Tapes" section in Chapter 6.

At the console, a mount request similar to the following prompts you to mount a tape on the tape drive:

```
====> OPERATOR.S_1 % SYSTEM_BACKUP.BACKUP <====
Please Load Tape
              (Kind of Tape => CHAINED_ANSI,
               Direction    => WRITING,
               Vol Set Name => BACKUP, 31-AUG-92 21:02:36 3)

Tape mounted and ready to read labels?
```

1. If you will be writing on a 9-track tape, select a 2,400-foot tape.

   If you will be writing on an 8-mm tape cartridge, select a tape with enough capacity to store all the data you want to save. See Table 15-6 for information about how to choose the appropriate length of tape.

   Write-enable and mount the selected 9-track tape reel or 8-mm tape cartridge. Ensure that the tape drive is online. (For more information, see Chapter 15.)

2. If you want to continue with the backup operation, respond to the question: `Is Tape mounted and ready to read labels?` on the operator's console by typing `yes` and pressing [Return]. (If you want to stop the backup operation, type `no` and press [Return]. Then see the "Refusing a Mount Request" subsection in Chapter 6.)

   If your system has both a 9-track and an 8-mm tape drive, refer to Chapter 15 for other prompts regarding the selection of the appropriate drive.

   After these optional prompts, the system should display information about the tape that you mounted, followed by a request for you to verify that the tape can be overwritten.

   If your operator's console does not display something similar to the following tape-mount information, see the "Handling Problems Encountered in Reading/Writing Tapes" section in Chapter 6.

   For example, if you are reusing an old backup tape, information such as the following is displayed on the console:

   ```
   ====> OPERATOR.S_1 % SYSTEM_BACKUP.BACKUP <====
   Info on tape mounted on drive 0 is:
      (Kind of Tape     => CHAINED_ANSI,
       Writeable        => TRUE,
       Volume Id        => 007701,
       Volume Set Name => BACKUP, 07-AUG-92 07:00:02 3)

   OK to overwrite volume? [YES]
   ```

3. If you want to overwrite the mounted tape, press [Return]. (If you decide not to overwrite the tape you mounted, type `no` and press [Return]. The system will re-display the mount request, which allows you to mount another tape.)

   If you selected `YES`, the operator's console displays something like this:

   ```
   What should the volume id handling mode be? [AUTO_GENERATE]
   ```

4. Choose a method of assigning a volume identifier to the tape by typing one of the following options and pressing [Return]:

   - `Auto_Generate`

     Automatically generates a new, sequential volume identifier for this tape and any subsequent tapes in the volume set. This option is the easiest choice for multivolume sets.

- `Use_Existing_Volume_Ids`

  Preserves the existing volume identifier that is already on the tape that you mounted. This option can be used only for the first of multiple tapes in a volume set.

- `Operator_Supplied`

  Allows you to enter a list of volume identifiers to be used on this tape and any subsequent tapes in the volume set. With this option, you can construct informative volume identifiers or assign a volume ID according to your local requirements.

5. Regardless of the choice made in the previous step, the system eventually responds by assigning a volume identifier and then displaying it. For example, if you specify Auto_Generate, a message similar to the following is displayed:

   ```
   Volume id of tape is now: 007803
   ```

   After the volume identifier has been assigned, the tape drive begins to write data onto the tape. If the system data is large enough to require additional tapes, you will be prompted for them. (For information about volumes that span more than one tape, refer to the "Writing a Continuation Volume" subsection in Chapter 6.)

   Caution: *If you are prompted to supply a volume identifier for the next tape, you must supply this volume identifier before changing tapes. The reason is that the volume identifier for the next tape must be written on the end of the current tape. If you manually unload and dismount the current tape and mount and load the next tape before answering this prompt, the volume identifier labels will be written in the wrong place on the wrong tape and the backup will not be recoverable.*

   Also, you should never manually rewind and/or dismount a tape when making a backup. The backup program will always automatically unload the tape at the right time. If you must manually unload a tape at any time when making a backup, something has gone wrong and you should restart the entire backup.

6. After all of the data has been written, the backup index is written.

   If you are using a 9-track tape drive, you will be prompted to dismount the data tape and mount the separate backup-index tape.

   If you are using an 8-mm tape cartridge, the backup index is automatically written on the current cartridge.

7. Dismount the tape when prompted.

The backup operation is now complete. It is recommended that you now run the Verify_Backup procedure to ensure this backup is complete and can be used to restore the system. (For details, see the "Verify_Backup Procedure" subsection, below.)

If you encountered any problems, see the "Handling Problems Encountered in Reading/Writing Tapes" section in Chapter 6.

SAFEGUARDING SYSTEM INFORMATION

In the event of a catastrophic system failure, backups are your insurance that the system can be recovered to its state at the time of the last backup. In cases of disk-drive corruption or failure, recovering from backup is the only alternative. Hardware and software upgrades also may require rebuilding the system from backup.

Merely taking a backup, however, does not guarantee that such recovery will be successful. Because the entire system is being restored, the system manager should take additional precautions to reduce the risk of not being able to recover from backup tapes.

To this end, Rational recommends these guidelines:

- Follow the recommended backup schedule listed in Table 7-1.
- Perform regular archives (either to tape or to another Rational system) of important project work (see the "Archiving" section, later in this chapter).
- Verify full backups regularly (ideally for each full backup, or at a minimum monthly). See the "Verify_Backup Procedure" subsection, below.
- Make and verify full *duplicate* backups before any scheduled hardware or software upgrade that requires rebuilding the system from backup. See the "Verify_Backup Procedure" subsection, below.

Verify_Backup Procedure

A set of full backup tapes can be verified using the Verify_Backup procedure as described below.

1. Enter the command:

   ```
   Verify_Backup;
   ```

   This command verifies that the backup tape(s) can be read, that the information on the tape(s) is complete, and that recovery is possible. Running this procedure takes about half the time of restoring the Environment from the same tapes.

   Standard mount requests (as described in the "Handling the Backup Tape-Mount Request" section, above) are issued for each tape in the backup set. If the backup is on 9-track tapes, the backup-index tape will be requested first. If the backup is on 8-mm tape cartridges, the backup typically will occupy only one tape.
2. Mount the first 9-track tape or 8-mm tape cartridge, described in Chapter 15.
3. Respond to the tape-mount request using the procedures described in the "Responding to the Mount Request" for reading a tape subsection in Chapter 6.

The backup index will be read first, followed by the data that corresponds to each disk volume.

On a 9-track tape drive, this means that the backup-index tape must be mounted first and then each data tape, in order. You will be prompted for each tape when needed by the Verify_Backup procedure.

On an 8-mm tape drive, this means that the first tape cartridge must be mounted first and then any additional tape cartridges, in order. You will be prompted for each tape when needed by the Verify_Backup procedure.

The output displayed on the screen should resemble the following sample. (Note that this backup was made on a machine with an 8-mm tape-cartridge drive and four disk volumes.)

```
Backup_Time: 27-JUL-92 19:02:16
  Backup_Version: 3
Exabyte_Recovery: TRUE

Positioning tape to Backup Index
```

```
... Skipping File: Space Info Vol 1
... Skipping File: Block Info Vol 1
        . . .

Processing Backup Index

... Reading File:  Vol Info             1 blocks
... Reading File:  VP Info              2 blocks
... Reading File:  DB Backups          11 blocks
... Reading File:  DB Processors        9 blocks
... Reading File:  DB Disk Volumes     35 blocks
... Reading File:  DB Tape Volumes     45 blocks

Backup_Time changed to 19-AUG-92 10:55:48
Number_of_Backups set to 309

Positioning tape to Backup Data
Processing Backup Data

... Reading File:  Space Info Vol 1   681 blocks
... Reading File:  Block Info Vol 1    51 blocks
... Reading File:  Block Data Vol 1  4810 blocks
        . . .

... Reading File:  Space Info Vol 4   668 blocks
... Reading File:  Block Info Vol 4   111 blocks
... Reading File:  Block Data Vol 4  5657 blocks

Successfully completed scan of backup tape.
```

If the message `Successfully completed scan of backup tape` does not appear after Verify_Backup has completed, you should clean the tape drive and run the Verify_Backup procedure again. If Verify_Backup still detects a problem, you should suspect that the backup or the tape is bad. Redo the backup operation and reverify the tape using Verify_Backup. If an error is still detected, contact your Rational support representative. (See Appendix D for information on how to obtain customer support.)

In addition to using the Verify_Backup procedure, you can check to see if your backup procedure is listed as the newest entry in the backup log, as described below.

System_Backup.History Procedure

The System_Backup.History procedure displays the dates and volume identifiers for a specified number of the most recent backups.

For example, to display information about each of the last ten backups, enter the following command (10 is the default):

```
System_Backup.History;
```

A display resembling the following appears for each full, primary, or secondary backup:

```
------------------------------------------------------------
!USERS.OPERATOR % SYSTEM_BACKUP.HISTORY     STARTED 5:18:41 PM
------------------------------------------------------------

Incremental Backup 301 Taken At 19-AUG-92 07:53:50
                                Based On 15-MAY-92 08:55:12
  Blue Tape Vol Name => BACKUP, 19-AUG-92 07:47:24 3
  Blue Tape Vol Id   => 061500
  Data Tape Vol Name => BACKUP, 19-AUG-92 07:47:24 3
  Data Tape Vol Id   => 061500
Full Backup 300 Taken At        15-AUG-92 08:55:12
  Blue Tape Vol Name => BACKUP, 15-AUG-92 08:40:59 3
  Blue Tape Vol Id   => 007102
  Data Tape Vol Name => BACKUP, 15-AUG-92 08:40:59 3
  Data Tape Vol Id   => 007102
          . . .
Full Backup 292 Taken At        18-JUL-92 08:57:04
  Blue Tape Vol Name => BACKUP, 18-JUL-92 08:53:48 3
  Blue Tape Vol Id   => 061700
  Data Tape Vol Name => BACKUP, 18-JUL-92 08:53:48 3
  Data Tape Vol Id   => 061700
```

Check this log to see if the volume ID assigned for your backup tape is listed as one of the entries. If the volume ID is not present in the log, then your backup has failed and you must repeat the backup procedure. Note that the backup-index tape was formerly called a "blue tape"; the two names are synonymous.

Tape.Examine_Labels Procedure

To check your tape more quickly (but less thoroughly) than with the Verify_Backup procedure, enter the following command:

```
Tape.Examine_Labels (Volume_Labels_Only => False);
```

A diagnostic message will appear in the current output whenever the command detects an inconsistency with any of the tape labels on an ANSI-standard labeled tape. A diagnostic message attempts to explain the inconsistency.

Off-Site Tape Storage

Rational recommends that you follow a regular schedule to ensure that a certain number of reliable tapes are stored safely off site. This schedule can be used in conjunction with the recommended system-backup schedule shown in Table 7-1 at the beginning of this chapter.

In addition, you should include a DFS backup tape with tapes stored permanently off site. (For making a DFS backup tape, see the "Making a DFS Backup Tape" subsection, later in this chapter.)

Weekly Backups (Retained for One Month)

Once each week, preferably the last business day of the week, make a full backup, label it, and retain it for at least one month.

Monthly Backups (Retained for One Year)

Once each month, preferably the last week of the month, follow the procedure below (instead of making a regular weekly backup):

1. Use an approved tape-cleaning kit to run one cleaning pass on the tape drive.
2. Make a full backup.
3. Run the Verify_Backup procedure on the full backup.
4. Label this backup and retain it for at least one year.

Quarterly Backups (Retained Indefinitely)

Once every three months, preferably the last week of the quarter, label what would be the monthly backup and retain it indefinitely.

DFS Backups

A disk file system (DFS) backup should be included with any archive or backup tape you have placed in permanent storage. This will allow you to load an old DFS before recovering a system backup in case your system is upgraded with a new DFS. The new DFS can make recovery of the earlier system backup impossible unless the old DFS backup is loaded first. (See the "Making a DFS Backup Tape" subsection, later in this chapter.)

Number of Tapes Needed

For an R1000 whose data can fit on one 8-mm tape, this scheme requires about 20 tapes plus four per year for the quarterly backups. More tapes will be needed for other tape operations. The daily tapes should wear out fastest, but they should still last for more than a year (52 passes each). You may want to replace the daily backup tapes annually with new tapes. The old tapes could be discarded or used for less-critical purposes.

Verification of Long-Term Backups

Once per year, run the Verify_Backup procedure on your quarterly backups. Once per year, run the Tape.Display_Tape procedure on your DFS backups.

RESTORING THE ENVIRONMENT FROM A BACKUP TAPE

This section is divided into two subsections: "Background Information" and "Restoration Procedure."

Background Information

Restoring the Environment from a backup tape is always done from the operator's console. Commands entered at the `Kernel:` prompt initiate the recovery process, which eventually displays a tape-mount request for the backup tape. After you mount a tape on the tape drive and respond to the subsequent prompts, the Environment is rebuilt from data on the tape. If another tape is required, you will be prompted at the operator's console.

Progress Messages

During restoration, progress messages appear on the operator's console. These messages report when the backup index has been located and when it is being processed; subsequent messages report when the backup data has been located and when it is being processed.

Impact on Current Environment

Restoring the Environment from backup tapes involves erasing the current Environment state and replacing it with the state that was recorded on tape. Erasing the current Environment state is a drastic step, and it eliminates any possibility of recovering the Environment without using backup tapes.

Tape Issues

The major tape issues are as follows:

- A backup made on a 9-track tape drive always requires a minimum of two tapes: a data tape and a backup-index tape.

  A backup made on 8-mm tape-cartridge drives usually requires only one tape, which contains both the data and the backup index, in that order.

- When you are prompted for the backup-index tape during restoration from 9-track tapes, you mount the 9-track backup-index tape first. The system searches the tape for the backup index, which it finds immediately.

  When you are prompted for the backup-index tape during restoration from 8-mm tape cartridges, you mount the first data cartridge tape first. The system searches this cartridge for the backup index. This involves scanning through all the data (if there is more than one cartridge, you will need to mount each in turn) until the backup index is found just past the last of the data on the last cartridge.

- After the backup index is found on a 9-track tape, it is read into the system. Then the system rewinds the tape, prompts you to dismount the backup-index tape, and prompts you to mount the first data tape.

  After the backup index is found on an 8-mm tape cartridge, it is read into the system. Then the system automatically rewinds the tape to the beginning and, if this cartridge includes all the data for the backup (which it typically does), automatically reads the system data into the system. If the data spans more than one cartridge, the system prompts you to dismount the current cartridge and mount the first tape cartridge.

  System backups made on 8-mm tape cartridges for very large systems may require multiple tapes. The backup operation writes to these tapes as if they were logically a single tape—that is, the backup index is written immediately following the data on the last tape. Consequently, restoration of multitape backups is a two-pass process. The tape-mount request first prompts you to mount each tape in sequence so that the tapes can be skipped through until the backup index is found. Then the tape-mount request prompts you to mount the first tape in the volume set. After the first data tape is mounted, the system begins to read the data and restore the Environment. Note that you cannot go directly to the last tape in the set.

- You can restore the Environment from a full, primary, or secondary backup. If you restore from a full backup, you will need to use tapes from only a single backup. For the other situations:

- A primary backup saves all the changed state since the last full backup and is therefore always based on a full backup. If you restore from a primary backup, you will need to load the primary backup first and then the full backup. A primary backup cannot be used by itself to restore the entire Environment.
- A secondary backup saves all the changed state since the last primary backup. You must load the secondary backup first, then the primary backup, and finally the full backup. Like a primary backup, a secondary backup cannot be used by itself to restore the Environment.

Restoration Procedure

The procedure described below summarizes how to restore the Environment from any kind of backup tape and assumes that your tape drive is online and that you mount the correct tape. If you encounter errors while restoring a system backup, refer to the "Handling Problems Encountered in Reading/Writing Tapes" section in Chapter 6.

To restore the Environment from any kind of backup:

1. Press [Control][Z] on the operator's console until the `Kernel:` prompt appears.
2. At the `Kernel:` prompt, enter the following command and press [Return]:

   ```
   Enable_Priv_Cmds
   ```

   The system responds with a message and then asks if you want to proceed:

   ```
   You are enabling a set of commands which must be used
   with extreme care.  They should be used only by
   knowledgeable support personnel.  These commands can
   easily crash/hang the machine; some can completely trash
   the state of the machine such that you must recover the
   machine from backup tapes.
   Proceed [FALSE]:
   ```

3. To proceed, enter `yes` or `true` at the prompt and press [Return].

   The system responds by requesting a password.

4. Enter the default password `secret` (or the valid password, if different) and press [Return].

   The system responds with the privileged prompt `*Kernel:`.

5. At the `*Kernel:` prompt, enter the following command and press [Return]:

   ```
   Go_Back_In_Time
   ```

   The system responds with a message warning you that the current Environment state will be destroyed if you proceed with this command, and a prompt is displayed, asking if you want to proceed:

   ```
   The purpose of this command is to trash the current state
   of this machine.  When the system is next booted, it will
   require that you build a new virtual memory system and
   recover from backup tape.
   Proceed [FALSE]:
   ```

6. At this prompt, type `yes` or `true` and press [Return] to proceed.

   The system responds with the following prompt:

   ```
   SNAPSHOT_NUMBER [0]?
   ```

   This prompt is asking you to confirm that the system should go back to "snapshot 0," the state before any snapshots.

Caution: *This step is irreversible and eliminates the possibility of recovering without the use of backup tapes.*

7. At this prompt, press [Return]. The system displays:

```
====>> CONFIGURATOR <<====
shutdown to go back in time
```

The system shuts itself down and begins the normal reboot process. The operator's console displays the usual boot messages and pauses at the following menu:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Inhibit R1000 CPU state save into Tombstone
    4 => Boot system, prompting for tape or disk

Enter option [RETURN will continue booting] :
```

8. Press [Return] to continue the process.

The system continues to boot. If the operator-mode keyswitch is set to Interactive, the following menu appears:

```
=====================================================
Restarting system after R1000 Processor Halt => Shutdown from
Environment

R1000 Crash Save done - tombstone file R1000_DUMP1 created.

>>> Notice: the OPERATOR_MODE switch is set to INTERACTIVE <<<
Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

Press [Return] to continue booting.

Messages appear on the operator's console until the system starts to rebuild virtual memory. At this point, the system departs from the usual boot process and requests instructions to build a new virtual-memory system:

```
====>> CONFIGURATOR <<====
starting diagnosis of configuration
recovery is needed
WANT TO BUILD NEW SYSTEM (YES/NO):
```

9. Enter yes and press [Return].

The system responds with the following question:

```
====>> CONFIGURATOR <<====
Setting free blocks to gc footprint takes about 15 minutes per
drive.
Want to set free blocks to gc footprint?
```

10. Enter no and press [Return].

The system displays more console messages, and then asks the following question:

```
====>> Recovery <<====
Recovery Is Needed, Should I fake it?
```

11. Enter no and press [Return].

    Answering *no* instructs the system to prepare for reading backup tapes. Answering *yes* bypasses the backup tapes. If you answer *yes,* you must begin the recovery process again, starting with step 1.

    The system requests the volume identifier of the backup index (that is, the first tape of the volume set):

    ```
    Please tell me Volume Id for Backup Index Tape:
    ```

12. If you do not know the volume identifier, press [Space Bar] [Return] and the system will accept whatever tape is mounted.

    A mount request for the specified volume is displayed:

    ```
    ====> SYSTEM % RECOVERY <====

    Please Load Tape
      (Kind of Tape     => CHAINED_ANSI,
       Direction        => READING)
    Is Tape mounted and ready to read labels?
    ```

13. If you have not already done so, mount the specified tape (for the mounting procedure, see Chapter 15) and answer the question:

    ```
    Is Tape mounted and ready to read labels?
    ```

    by typing yes and pressing [Return].

    The system responds by displaying information about the tape you mounted and checking the volume identifier of the mounted tape against the volume identifier you specified in step 12.

    If the tape you mounted matches the tape you specified in step 12, the system displays the following prompt:

    ```
    OK to read volume? [YES]
    ```

    This prompt is asking for permission to read the tape.

14. At this prompt, press [Return].

    After you have answered the prompt, the backup recovery process begins.

15. The procedure at this step depends on whether you are restoring from 9-track tapes or 8-mm tape cartridge(s).

    - If you are restoring from 9-track tapes, you will be prompted when to remove the backup-index tape and mount the first data tape.
    - If you are restoring from 8-mm tape cartridge, and the backup required just one cartridge, the system will automatically skip past all the data to the backup index, read in the backup index, rewind the tape, and then read in the data.

      If the backup required more than one 8-mm cartridge, then the backup index is located at the end of the *last* tape cartridge of the volume set. The console prompts you for each tape cartridge in the set and skims through each tape until the backup index is found. After reading the backup index, the system prompts you to mount the first cartridge of the volume set. If more tapes are required during the backup recovery process, you will be prompted for them.

16. If you are restoring from a secondary or primary backup:

    - After all secondary backup tapes are read, you are prompted for the first data tape of the corresponding primary backup.
    - After all primary tapes are read, you are prompted for the first data tape of the corresponding full backup.

Note that you will not need the backup index from the full backup or the primary backup.

After all the data has been read into the system, this message appears on the console:

```
====>> Recovery <<====
Recovery Is Complete
Garbage Collection can't run until the machine is rebooted
Rebooting to enable Garbage Collection
```

The system now reboots itself.

ARCHIVING

Archiving refers to Environment facilities for writing one or more objects onto tape and then restoring those objects to the Environment. In contrast to system backup, which saves an image of the entire Environment, archiving allows you to save individual objects such as worlds, Ada units, subsystem views, and so on. Restoring from system-backup tapes returns the entire Environment to a previous state.

Because of its flexibility, archiving is especially useful for recovering accidentally expunged objects. Therefore, your installation may decide to archive important worlds or projects on a regular basis to supplement the regular system backups. Note that archiving should not be used *in place* of the recommended system backups, because the archive format is not suitable for the restoration of the entire Environment—for example, in case of disk failure. Archiving takes considerably longer than system backups, so regularly archiving the entire Environment is not practical.

Archiving is described in the Library Management (LM) book of the *Rational Environment Reference Manual.* Also see the "Archive Commands" subsection at the end of Chapter 4 for relevant details about archiving and access control.

Following a discussion about archiving schedules, the rest of this section describes procedures for responding to tape-mount requests for archives.

Note: *For an overview of access issues related to package Archive, see subsections "CMVC and Subsystems" and "Archive Commands" in Chapter 4.*

Recommended Archiving Schedule

Rational recommends the archiving schedule shown in Table 7-2.

Table 7-2 Recommended Archiving Schedule

Type of Data	Frequency Suggested
General user data (!Users)	Every week
Active projects (!Projects)	Every week and at project milestones
Critical projects (!Projects)	Every week and daily incrementals

Archive tools can be written to send data automatically to disk rather than to tape. For example, you could archive to libraries named "Monday," "Tuesday," and so forth, if the total amount of data is not excessive. Archive does compress data compared to the space it normally takes up on disk. Other automated tools may be developed to perform archives automatically to other systems on the network.

Handling the Mount Request for Archive.Save

The procedure described below summarizes how to handle the tape mount initiated by the Archive.Save procedure. If you encounter errors while responding to this tape mount, refer to the "Handling Problems Encountered in Reading/Writing Tapes" section in Chapter 6.

Shortly after an Archive.Save command is entered, a mount request such as the following appears on the operator's console, prompting you to mount a tape on the tape drive:

```
====> BOBR.S_1 % ARCHIVE.SAVE <====
Please Load Tape
  (Kind of Tape     => CHAINED_ANSI,
   Direction        => WRITING,
   Volume Set Name => *DATA_SHORT*)

Is Tape mounted and ready to read labels?
```

1. If you are making this archive on a 9-track tape drive, write-enable and mount a tape on the tape drive. Set the tape density. Ensure that the drive is online.

   If you are making this archive on an 8-mm tape-cartridge drive, write-enable and mount a tape cartridge on the tape drive. Close the tape-drive door.

2. Answer the following question on the operator's console:

   `Is Tape mounted and ready to read labels?`

   by entering `yes` and pressing [Return].

3. If you have a dual-tape system, you may have to answer additional prompts. The system responds by displaying information about the tape you mounted.

   For example, if you are using a blank tape, messages like this are displayed:

   ```
   ====> OPERATOR.S_1 % ARCHIVE.SAVE <====
   Info on tape mounted on drive 0 is:
     (Kind of Tape     => EMPTY,
      Writeable        => TRUE)

   Tape will be overwritten in GCR_6250CPI. If not correct,
   change at drive.
   OK to overwrite volume? [YES]
   ```

   If you are using an 8-mm tape-cartridge drive, the line about tape density will not be displayed.

4. If you do not want to overwrite the tape you mounted, enter `no` and press [Return]. The mount request is redisplayed and you can mount another tape.

   To overwrite the mounted tape, press [Return] and continue with the next step.

5. The system displays the following prompt, so you can assign the tape a volume identifier:

   `What should the volume id handling mode be? [AUTO_GENERATE]`

   Enter one of the following and press [Return], or simply press [Return] to choose the default displayed in the prompt:

   - `AUTO_GENERATE` automatically generates a new, sequential volume identifier for this and subsequent tapes in the volume set. This option is the easiest choice for operators.

- OPERATOR_SUPPLIED allows you to specify a list of volume identifiers to be used on this tape and any subsequent tapes in the volume set. With this option, you can construct informative volume identifiers.
- USE_EXISTING_VOLUME_IDS can be specified if you mounted a tape that already has a volume identifier. (This option is not possible in the example in step 3.) This option preserves the existing volume identifier, and it can be used only for the first of multiple tapes in a volume set.

Regardless of the choice made in the previous step, the system eventually responds by assigning a volume identifier and then displaying it. For example, if you specify AUTO_GENERATE, a prompt such as the following appears:

```
Volume id of tape is now: 005460
```

After the volume identifier has been assigned, the Environment begins writing the data file onto the tape. If another tape is required, the system will prompt for it.

Handling the Mount Request for Archive.Restore

The procedure described below summarizes how to handle the tape mount initiated by the Archive.Restore procedure. If you encounter errors while responding to this tape mount, refer to the "Handling Problems Encountered in Reading/Writing a Tape" section in Chapter 6.

Shortly after an Archive.Restore command is entered, a mount request appears on the operator's console. Note that the mount request itself does not identify the tape that is to be mounted. The user who entered the Archive.Restore command must indicate the tape that is required (check your Environment terminal in case a user sent you a message requesting a particular tape).

The following steps show the prompts and messages that result from entering R1000_Long format in the Archive.Restore command. (Even on an 8-mm tape cartridge, R1000_Long format puts the index on a separate tape.) If the user specified R1000 format instead, you will need to read only one tape.

1. A mount request is displayed on the console:

```
====> MAM.S_1 % ARCHIVE.RESTORE <====
Please Load Tape
  (Kind of Tape    => CHAINED_ANSI,
   Direction       => READING,
   Volume Set Name => *INDEX*)

  Is Tape mounted and ready to read labels?
```

   Note that the requested volume-set name is *INDEX* when the command specifies R1000_Long format (that is, the index data is on a separate tape). If R1000 format is specified, the requested volume-set name is *DATA_SHORT*.

   Mount the appropriate tape on the tape drive and make sure the drive is online. Density is set automatically to match the tape.

2. Respond to the console prompt by typing yes and pressing [Return].
3. The system responds by displaying console information about the tape you mounted and asking you for permission to read the tape:

```
Info on tape mounted on drive 0 is:
  (Kind of Tape     => CHAINED_ANSI,
   Writeable        => TRUE,
   Volume Id        => 005464,
   Volume Set Name => *INDEX*)
OK to read tape? [YES]
```

Press [Return] to continue.

4. After the backup index has been read, the system unloads the index tape and asks for the first data tape to be loaded. Dismount the index tape and repeat steps 1 through 3 for each data file tape, continuing until all the tapes are read.

OTHER TAPE OPERATIONS

Other tape operations that system managers may need to perform include:

- Making a DFS backup tape
- Making a SIMS log tape
- Using the Tape.Read and Tape.Write procedures
- Transferring information between systems

Each of these operations is described in separate subsections below.

Making a DFS Backup Tape

If you are making a backup for off-site storage, you should also make a backup of the disk file system (DFS) that will accompany this tape in permanent storage. This will allow you to load an old DFS before recovering a system backup in case your system is upgraded with a new DFS. The new DFS can make recovery of the earlier system backup impossible unless the old DFS backup is loaded first. Once you have made a DFS backup, this tape should be included with any backup tape you have placed in permanent storage (see the "Off-Site Tape Storage" subsection, earlier in this chapter).

To make a DFS backup:

1. Make sure the operator-mode keyswitch is in the Interactive position.
2. Stop the Environment using the procedure described in the "Normal Environment Shutdown" section in Chapter 2.

   As soon as the shutdown completes, the system automatically starts to boot. Because the operator-mode keyswitch is in the Interactive position, the boot procedure pauses at the following menu:

```
Restarting R1000-200 August 21st, 1992 at 09:50:46

Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Inhibit R1000 CPU state save into Tombstone
    4 => Boot system, prompting for tape or disk

Enter option [RETURN will continue booting] :
```

3. Press [Return] to continue booting.

4. After displaying some more console messages, the system pauses at another menu:

```
R1000 Crash Save done - tombstone file R1000_DUMP1 created.

>>> Notice: the OPERATOR_MODE switch is set to INTERACTIVE <<<
Options are:
  1 => enter CLI
  2 => make a CRASHDUMP
  3 => run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

   Enter `1` and then press return to enter the CLI.

   Your operator's console should now display the `CLI>` prompt.

5. Mount a tape.

   For the mounting procedure for either a 9-track tape drive or an 8-mm tape-cartridge drive, see Chapter 15.

   For the 9-track drive, use 2,400-foot tapes.

   For the 8-mm tape-cartridge drive, any of the tapes listed in Table 15-6 should have sufficient data capacity to store all of the DFS data on one tape.

6. Enter the `backup` command at the `CLI>` prompt:

```
CLI> backup
```

   There is no mount request after executing this command—the tape operation begins immediately.

   A message similar to the following appears:

```
Writing tape marks and repostioning tape takes about 45
seconds.
Elapsed time is 00:04:44
total of 3243 files dumped.
```

7. Wait for the operation to complete (the `CLI>` prompt reappears) and for the tape drive to rewind the tape. Then dismount the tape.
8. If your system's operator-mode keyswitch was originally in the Automatic position, make sure you move the keyswitch back to this position.
9. To reboot the system, enter the `boot` command and press [Return]:

```
CLI> boot
```

   The following prompt appears:

```
Enter name of configuration to boot [STANDARD] :
```

10. Press [Return] to continue booting.

Making a SIMS Log Tape

As system manager, you may instruct your site's end users to report problems from their terminals using the Request program. If this is the case, then you may want to edit the SIMS log and record this information onto tape. This tape can then be sent to the appropriate support personnel for analysis. (For details about how to submit a support request, see Appendix D.) To edit the SIMS log file, you must use the Review tool.

The Review tool is invoked by executing the Review command in a command window. The program brings up a text window that shows the automatically generated template with defaults.

Assuming that the system users have submitted several requests or problems to be forwarded to Rational, the system manager typically follows these steps daily or weekly:

1. Start the Review program by executing the Review command in a command window:

```
Review (Sims_Area => "!machine");
```

A display and command prompt similar to the following appears:

```
SIMS - Rational Customer Support Information Management System
============================================================

Company Name : New Company
System ID  # : 912369
Machine Name : Gator
Local Site   : new_site
User Name    : GERIB
Configuration: D_12_5_0
Checking for User Requests/Problems to Load =>
Processing input files from:
!machine.sims.to_rational.receiving.@
  Found no new user input files to process.

  Command: (Display, Search, Report, Send, Forward, Quit,
Menu) : [Menu] :
```

2. To see a menu that displays all of the available options, press [Promote]:

```
Support Requests Management
=======================================
 1. Display a Customer Support Request   2. Edit an existing Support Request
 3. Enter a new Problem or Request       4. Detailed entry of a new Request
 5. Search for a specific request        6. Continue last search
 7. Current request display              8. Next request display
 9. Prior request display               10. Log requests to print log
11. Newly opened requests search        12. Show last request added to file
13. Print current request to log        14. Help information overview
15. Report or Display all requests      16. Send existing request to Rational
17. Close the current request           18. Forward all 'to_be_sent' to Rational
19. Delete the current request          20. Initialize a new master file
21. Menu display of all commands        30. Quit this application
[Select a command by number or by unique portion of command name.]
[Or '?' for new menu, Or use standard R1000 window & editing.    ]
[Or for record editing:   Up= ^   Top= ,   Select= .   Exit= /   ]

  Command: (Display, Search, Report, Send, Forward, Quit, Menu):[menu]:
```

3. To search for any newly opened user requests or problems, type `11` and press [Promote].
4. Review new user requests or problems, deciding whether to handle them locally or forward them to Rational.
5. If the request or problem can be handled locally, change its status to Pending to indicate you are working on the problem. When the issue is resolved, update the report, close it, and give a listing to the system user, using the Print command.

6. If the request is to be forwarded to Rational, you may want to add some detail or change the priority. As you review each of these reports, invoke the Send com-
   mand, which updates the status to To_Be_Sent.
7. Repeat the process as desired until all open reports are processed.
8. At a selected interval, invoke the Forward command to collect and transfer all To_Be_Sent problems to tape for submission to Rational. The status is updated to Sent on all such reports.

For help at any point, type the ? command to see the available commands or choices. In addition, type the ? command when editing or entering a field in a request to show the valid options allowed.

Using the Tape.Read and Tape.Write Commands

Several operations involve reading from or writing to a tape using the !Commands-.Tape.Read and Tape.Write commands. For example, a user who needs to transfer a program to another computer from an R1000 can write the program onto a tape using the Tape.Write procedure. Similarly, a user desiring to read the contents of a tape into an Environment library can initiate the process by entering the Tape-.Read procedure. The other system must have a tape-drive model that is identical to the tape drive on the R1000, or it may not be possible to transfer the data between machines.

Tape operations using the Tape.Read and Tape.Write commands are covered in further detail in package Tape of the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

The tape-mount and tape-dismount procedures for reading and writing system information using the Tape.Read and Tape.Write commands are the same as procedures for backup and archive. Note that users may be required to fill in information about tape formats, which is not required with either backup or archiving operations. See the following subsection for more information about transferring information between systems.

Transferring Information between Systems

This subsection briefly describes procedures you can use when reading and writing tapes to or from the R1000 and other machines.

Between the VAX and the R1000

To create a tape on a VAX for the R1000, use the VAX/VMS Copy command (see your VAX/VMS documentation). Once the tape has been written, the tape can be read on the R1000 with the Tape.Read command with an Options parameter specifying "format=vax/vms."

To write a tape on the R1000 for VAX/VMS, use the Tape.Write command with an Options parameter specifying "format=vax/vms."

Between Other Machines and the R1000

To read or write tapes for tar, MVS, or other machines and the R1000, see the "Tape Utilities" tool in the *Rational Software Library Catalog.*

8

System and Hardware Reports

The !Tools.System_Availability subsystem has been provided to allow you to generate reports on system usage, problems, and daemons. Specifically, it allows you to generate the following eight reports:

- Availability: System uptime and downtime by class
- Usage: Number of users on the system, at hourly intervals
- Disk: Disk space used each day
- Devices: Errors from devices such as disks, tape, and memory
- Daemons: Sizes and schedules of daemons
- Outages: System outages and reasons for them
- Trouble: Potential trouble areas]
- Everything: All of the above reports

This chapter consists of two sections. The first section specifies the procedure for generating any of the eight reports listed above, the information these reports contain, and an example of each report as it appears on the screen. The second section describes two additional reports on hardware that you can generate.

SYSTEM REPORTS

To generate any of the system reports, use the System_Report.Generate command. This command has the following specification:

```
procedure Generate
     (Report_Type    : Report_Class := System_Report.Everything;
      Start_Time     : String       := "";
      End_Time       : String       := "";
      Log_Directory  : String       := "!Machine.Error_Logs");
```

The Report_Type parameter allows you to specify what kind of report you want to generate. You must specify one of the report types in the Report_Class enumeration type shown below:

```
type Report_Class is
              (Availability, -- Uptime/downtime by classes
               Usage,        -- Per half hour, # users, etc.
               Disk,         -- Used disk space each day
               Devices,      -- Disk, Mem, Tape, etc. errors
               Daemons,      -- Daemon state sizes and times
               Outages,      -- System outages and reasons
               Trouble,      -- Potential trouble areas
               Advice,       -- Advice on cleaning things up
               Everything,   -- All reports
               Tape_Mounts); -- Tape processing for backups
```

If you use the default value for the Report_Type parameter, the report generated will be for everything.

The Start_Time parameter allows you to specify the time and day for the beginning of the report you are generating. For example, to generate a report beginning at 10 A.M. on August 16, 1992, use the value "08/16/92 10:00". To generate a report beginning at 10 P.M. that day, use the value "08/16/92 22:00". A null value for this parameter (the default) generates a report for the earliest time and date available.

The End_Time parameter allows you to specify the time and day for the end of the report you are generating. For example, to generate a report ending at 10 A.M. on May 8, 1992, use the value "05/08/92 10:00". To generate a report ending at 10 P.M. that day, use the value "05/08/92 22:00". A null value for this parameter (the default) generates a report for the latest time and date available. When specifying values other than the defaults, specify a range for the Start_Time and End_Time parameters that is within the range of log files that exist on the machine.

The Log_Directory parameter specifies the location of the log files from which the report will be generated. Unless you have moved your log files to another library, you do not need to change this value.

Eight of the reports are described in separate sections below. The reports generated using the Advice and Tape_Mounts parameters are not described because these reports are either incomplete or not applicable.

System Availability

To generate a report for system availability from July 16 through August 8, 1992, execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
       (System_Report.Availability, "07/16/92", "08/08/92");
```

Executing this command produces a report similar to the following:

```
      Time              Length       Cause                Comments
==================    ========    =======    ================================
92/07/16 14:25:19      00:00       None
92/07/21 16:27:53      01:16      Release    by OPERATOR.FMJ Installation of
                                              MC68020_HP_Unix Release6_2_2
92/07/22 10:08:21      00:54      Release    by OPERATOR.FMJ RDF 2167a Install

92/07/24 21:52:24      00:51      Release    by OPERATOR.FMJ Mail
                                              Release11_5_0 Install
92/08/06 13:28:44      2/00:20     None


System Availability Statistics
Total outages                                =    5
Total downtime                               =    2/03:20
Total report time                            =    22/21:34
Downtime due to system problems              =    03:00
Downtime due to planned operations           =    2/00:20
Total uptime fraction                        =    90.7
System availability fraction                 =    99.5
```

The information generated in each of the fields is described below.

- Total outages: Indicates the number of times that the system was taken down.
- Total downtime: Indicates the number of hours and minutes that the system was down.
- Total report time: Indicates the number of hours and minutes covered by this report.
- Downtime due to system problems: Indicates the amount of time that the system was down because of system problems.
- Downtime due to planned operations: Indicates the amount of time that the system was down for planned operations.
- Total uptime fraction: Indicates the percentage of time that the system was up.
- System availability fraction: Indicates the amount of time that the system was available to be up (that is, the amount of time during which there were no system problems that would have prevented it from being available). Note that, because of scheduled maintenance, this can be larger than the total uptime fraction.

System Usage

The system-usage report indicates the number of users logged in for each day (00:00 to 24:00 hours) of the specified time period. A period (.) indicates that no one was logged in. Numbers indicate the maximum number of people logged in during that hour. A dash (-) indicates a period when the system was unavailable. An asterisk (*) indicates that the disk daemon was running. The display is divided into months. This system report is useful for a tabular view of usage and disk daemon execution.

For example, to generate a report for system usage from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Usage, "07/16/92", "08/08/92");
```

Executing this command would produce a report similar to the following:

```
                                Monthly Usage Summary
Time    00    02    04    06    08    10    12    14    16    18    20    22 23
=============================================================================
07/16                                                   1   ---   .  .  .  .  .  .
07/17    .  .  .  ***  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
07/18    .  .  .  ***  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .
07/19    .  .  .  ***  .  .  .  .  .  1  .  .  1  1  1  1  1  1  1  .  .  .  .  .
07/20    .  .  .  ***  .  .  .  .  1  2  1  1  1  2  2  2  1  1  .  .  .  .  .  .
07/21    .  .  .  ***  .  .  .  .  1  1  1  3  2  2  3  3  3  3  2  2  .  .  .  .
07/22    .  .  .  ***  .  .  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  .  .
07/23    .  .  .  ***  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  1  1  .
07/24    .  .  .  ***  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  .  .  .  .
07/25    .  .  .  ***  .  .  .  .  .  .  .  .  1  1  1  .  .  .  .  .  .  .  .  .
07/26    .  .  .  ***  .  .  .  .  2  3  3  3  3  3  3  3  3  3  3  2  .  .  .  .
07/27    .  .  .  ***  .  .  .  .  4  4  4  3  4  5  4  4  4  4  3  3  2  3  3  1
07/28    1  1  1  *1*  1  1  1  2  4  6  6  6  6  6  6  6  6  5  5  3  ---  1  1  .
07/29    .  .  .  ***  .  .  .  .  2  4  4  4  4  4  3  3  3  3  2  1  1  .  .  .
07/30    .  .  .  ***  .  .  .  .  3  3  5  4  5  4  5  4  4  4  3  3  1  1  1  1
```

```
                                  Monthly Usage Summary
Time    00     02     04     06     08     10     12     14     16     18     20     22 23
=============================================================================================
08/01   1  1   1 *1*  1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1  1
08/02   1  1   1 *1*  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  1
08/03   1  1   1 *1*  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  1
08/04   1  1   1 *1*  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1  1
08/05   1  1   1 *1*  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1  1
08/06  *1* 1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   2  2
08/07  *2* 2   2   2   2   2   2   2   3   4   4   4   3   3   3   3   3   3   2   2   1   1   1  1
08/08  *1* .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
```

Daily Disk-Space Usage

To generate a report for disk-space usage and disk-space collection from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Disk, "07/16/92", "08/08/92")
```

Executing this command would produce a report similar to the following:

```
  Date          Vol 1              Vol 2              Vol 3              Vol 4
========  ================  ================  ================  ================
92/07/16
92/07/17  107 -> 99 00:12    10 -> 9 00:02     13 -> 12 00:02     11 -> 10 00:02
92/07/18  231 -> 205 00:28   82 -> 67 00:11    92 -> 86 00:16     61 -> 56 00:10
                                  . . .
92/08/07  181 -> 134 00:22   99 -> 85 00:11    75 -> 70 00:09     87 -> 78 00:07
92/08/08  191 -> 145 00:25  131 -> 85 00:14   105 -> 81 00:12    151 -> 118 00:12
```

The `Date` column indicates the date(s) for which the report is generated. Each of the other columns shows three items of information: the first three-digit number (for example, `107` in the `Vol 1` column for `92/07/17`) shows the amount of space used on that volume, in megabytes; the second number (for example, `99` in the `Vol 1` column for the same day) indicates the amount of space used after disk collection was run; the last number indicates the amount of time, in minutes, taken to collect space on the volume (for example, `00:12` in the `Vol 1` column for `92/07/17` indicates that it took 12 minutes to collect space on that volume for that day).

Device Errors

To generate a report for device messages and errors from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Devices, "07/16/92", "08/08/92");
```

Executing this command would produce a report similar to the following:

```
Device Events
Total Disk messages      =  0
Total Tape messages      =  6
Total Memory messages    =  0
Total Ethernet messages =  2

Log messages concerning Ethernet errors:
----------------------------------------
92/08/16 17:37:30 Ethernet is up.
92/08/28 20:47:35 Ethernet is up.
```

The first section of the report summarizes the number of messages generated for each of the devices. (Note that the number of messages generated does not necessarily match the number of errors; messages are generated for other reasons.) The second part of the report shows messages generated because of device errors.

The report shows that there were no disk messages, six tape messages, no memory messages, and two Ethernet messages. All of the messages except the Ethernet messages were normal, so no log message is shown for them.

Daemon Sizes and Times

To generate a report for daemon sizes and times from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
           (System_Report.Daemons, "07/16/92", "08/08/92");
```

Executing this command would produce a report similar to the following:

Date	Ada	File	Action	Directory	DDB
92/07/16			55 -> 55		
92/07/17	549 -> 373	584 -> 446	56 -> 56	1140 -> 720	
92/07/18	816 -> 710	2250 -> 1439	56 -> 56	4241 -> 2448	
92/07/19	716 -> 713	1451 -> 1444	479 -> 479	2468 -> 2463	
		. . .			
92/08/06	1102 -> 1102	1998 -> 1998	96 -> 96	3404 -> 3403	177 -> 168
92/08/07	1144 -> 1102	2009 -> 1998	185 -> 185	3417 -> 3403	177 -> 168
92/08/08	1173 -> 1130	2753 -> 2048	185 -> 185	5374 -> 3495	177 -> 168

For each disk, this display indicates the amount of used space after garbage collection has finished that day.

Date	Volume 1	Volume 2	Volume 3	Volume 4
92/07/16				
92/07/17	99146	8731	11858	9935
92/07/18	204529	66786	85811	55834
92/07/19	207617	77328	87130	69070
		. . .		
92/08/07	134235	84966	69972	77667
92/08/08	144593	84865	81455	117966

This report indicates the size in pages, before and after compaction, of each client run during the day shown in the Date field.

System Outages and Reasons

To generate a report for system outages from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Outages, "07/16/92", "08/08/92");
```

Executing this command would produce a report showing the times at which the system was taken down (the Time field), how long the system was down (the Length field), why the system was taken down (the Cause field), who took the system down, and the comments they entered (the Comments field). This report is identical to the system-availability report as described earlier.

Potential Trouble Areas

To generate a report for trouble areas from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Trouble, "07/16/92", "08/08/92");
```

Executing this command would produce a report similar to:

```
Exception conditions in log - may indicate serious problem
------------------------------------------------------------
92/07/16 14:19:27 !!! Transfer.Carrier_Map_Access
                      Initial_Map_Load_Failed Carrier_Map.
                      Load failed with unhandled exception:
                      !Io.Io_Exceptions.Name_Error (nonexistent
                      object) raised at PC = #A2E00D, #2BF
92/07/16 14:19:27 !!! not_found: !machine.transfer.GLOBAL
                  !!! not_found: !machine.transfer.LOCAL
92/07/16 14:28:00 !!! Program ERROR Initialize_Design_Facilities
                      is not part of !Machine
                      . . .
92/07/22 13:35:33 !!! CE_Initialization Core_Editor
                      XSUN4_COMMANDS could not be Installed.
92/07/22 13:35:33 !!! CE_Initialization Core_Editor XSUN4 not
                      installed; terminal keybindings not made.
92/07/24 21:42:11 !!! not_found: !machine.transfer.GLOBAL
                  !!! not_found: !machine.transfer.LOCAL
92/07/24 21:42:11 !!! Transfer.Carrier_Map_Access
                      Initial_Map_Load_Failed Carrier_Map.
                      Load failed with unhandled exception:
                      !Io.Io_Exceptions.Name_Error (nonexistent
                      object) raised at PC = #A2E00D, #2BF
```

```
Operations on Users
-------------------
92/07/21 16:25:00 --- Predefined_Users Created_user : Rational
92/07/23 14:07:31 --- Operator Created_User cdf_trng_master
92/07/25 10:21:17 --- Operator Created_User fmj
92/07/26 10:26:02 --- Operator Created_User mrb
92/07/26 10:26:34 --- Operator Created_User sjl
92/07/26 10:26:49 --- Operator Created_User sck
                  . . .
92/08/07 09:16:08 --- Operator Created_User srp
92/08/07 09:17:17 --- Operator Created_User RH
92/08/07 09:17:33 --- Operator Created_User SRM

Other Events
------------
```

This report shows exception conditions in the error log that could indicate potential problems. It also shows operations on user accounts and other events of possible interest.

Everything (All Reports)

You can also generate a single report that contains all of the reports described above.

To generate this report from July 16 through August 8, 1992, you would execute the following command from a command window or from the operator's console command interpreter:

```
System_Report.Generate
          (System_Report.Everything, "07/16/92", "08/08/92");
```

HARDWARE REPORTS

It is sometimes useful to generate reports about your system's hardware. Two such reports, one summarizing manufacturer's bad blocks for disks and another summarizing board information, are described below. Note that the known bad blocks found on disks during manufacturing are stored in a table; the system retargets any data that would be written to these areas. Any additional bad blocks that the system detects during system operation are also retargeted.

Bad Blocks

The bad-blocks report shows a summary of bad-block information for disks. It can be produced by executing the command:

```
System_Report.Show_Bad_Blocks;
```

The result of executing this command is a report similar to the following:

```
Manufacturers bad blocks, Volume = 1, Count = 209
       142        1A9        2CB        564
       5D4        744        924        B04
       CE4        EC4        F0C       1373
      1445       184B       1A3D       288C
                  . . .
     57F3A      5963D      5A3DE      5A85A
     5B306      5BCCD      5DB78      5EC62
     5FA40      5FAD6      60B94      61B74
     622D9

Manufacturers bad blocks, Volume = 2, Count = 245
       3C9        445        509        9E6
       E84        E89        F03       102F
      10F7       15BE       19F4       1A42
      1C23       1EAF       2799       2CDA
                  . . .
     5E11F      5E774      5EC89      5EFB1
     5F745      60023      600FC      60B76
     60BE3      6188B      6199D      619A9
     61D09

Manufacturers bad blocks, Volume = 3, Count = 434
        6C        264        62B        804
       821        9D2        A18        BE8
       BF7        D41        D75        F85
      1514       1837       1BA2       207D
                  . . .
     5C002      5C9F2      5D170      5E0CD
     5E0D9      5ED60      5F507      60204
     6115C      6176D      6226A      62622
     62802      629E2

Manufacturers bad blocks, Volume = 4, Count = 467
       227        44B        794        9C0
       D89       1037       116C       1244
      13BA       142B       1437       1684
      17DE       1CAD       1E7B       1FC7
     5DA39      5DE56      5DF52      5E289
     5E3C8      5E3CB      5EC12      5EDE6
     5F6A3      60B30      60F10      62006
     6268A      62744      62A06
```

Machine Information

The machine-information report shows serial numbers and board information for the machine. If a board is not part of the system configuration, this information is replaced by zeros. To produce this report, execute the following command:

```
System_Report.Show_Machine_Information;
```

Executing this command produces a report similar to the following:

```
Machine Id = 444990

R1000 Series 400 System

Board                  Part#   Serial#    Art Rev   ECO Level  Build Date
---------------------- ------- ---------- -------- ---------- -------------
Sequencer              45      4432       1        2          92/05/16
Field Isolation Unit   44      4124       1        1          92/02/25
IOC                    9       0          2        1          92/01/01
Type                   43      4401       2        2          92/05/16
Value                  42      4531       1        1          92/04/03
32 MB Memory           47      4483       2        4          92/04/23
32 MB Memory           47      4470       2        4          92/04/23
ReshA                  41      4350       2        7          92/05/21
CMC controller serial number = 629
```

9 Sending Messages

Often, when performing system-manager functions, it is necessary to communicate with the users logged into a machine. You can broadcast messages to all users or to individual users. Messages appear in the recipient's message window shortly after you send them.

System information of less immediacy can be posted to each user in the form of a *daily message*. The daily message typically is displayed in its own window each time a user logs in. For example, you can use the daily message to announce scheduled system downtime several days in advance.

Some messages—for example, warning messages for shutdowns or snapshots—are broadcast automatically by the relevant procedures (see "Executing the Schedule_Shutdown Command" in Chapter 2 and "Snapshot Warning Message" in Chapter 5).

BROADCASTING TO ALL USERS

To broadcast a message to all users:

Enter the !Commands.Message.Send_All procedure in a command window or from the operator's console command interpreter, supplying the desired message. For example:

```
Message.Send_All ("Who borrowed my reference manual? -sjl")
```

The message appears in the message window of each logged-in terminal. Users who are not logged in at the time the message is broadcast never receive it (it is not queued for eventual login).

SENDING TO A SINGLE USER

To send a message to an individual user:

Enter the Message.Send procedure in a command window or from the operator's console command interpreter, supplying the desired username and message. For example:

```
Message.Send ("Anderson","Please return my reference manual -sjl")
```

CREATING A DAILY MESSAGE

You can use the daily message to make general system information available to all users. The text of the daily message is kept in the !Machine.Editor_Data.Daily_Message file.

When a user executes the What.Message procedure, the system displays the text in !Machine.Editor_Data.Daily_Message. Typically, the What.Message procedure is part of the default login procedure, !Machine.Release.Current.Commands.Login, so the daily message appears whenever users log in. You may want to encourage users writing customized login procedures to be sure to call What.Message.

If you have added a System_Login procedure to !Machine, you can send messages unconditionally by adding a call to What.Message to !Machine.System_Login (eliminating the need to add this call to each customized login procedure). Refer to the "Systemwide Login Procedure" subsection in Chapter 3 and the "Preparing Daily Messages in Advance" subsection later in this chapter for more information about the System_Login procedure.

To write the text of the daily message:

1. Press [Create Text] and enter `!Machine.Editor_Data.Daily_Message` at the prompt:

   ```
   Text.Create (Image_Name => "!Machine.Editor_Data.Daily_Message",
                Kind       => Text.File);
   ```

2. Press [Promote] to edit the message.

   If no daily message previously existed, this brings up a new, empty file. Otherwise, it displays the current daily message. In either case, edit the file to contain the text you want.

   If a daily message already exists, edit the same file to change the message. Press [Prompt For] followed by [Edit], and enter `!Machine.Editor_Data.Daily_Message` at the prompt:

   ```
   Common.Edit (Name => "!Machine.Editor_Data.Daily_Message");
   ```

3. Press [Promote] to save the message.
4. Test the display of the message by executing the following command:

   ```
   What.Message;
   ```

Note that the ACL for the !Machine.Editor_Data.Daily_Message file should include Public=>R or Network_Public=>R. Like many other Environment commands, the What.Message procedure also can be used from the operator's console command interpreter; however, the message can only be viewed, not edited.

Displaying Text from Another File

You can use the What.Message procedure to display text from other files besides the default !Machine.Editor_Data.Daily_Message. For example, you could create a file called Bulletin_Board that users themselves could update and check periodically for new announcements or other information. Although such a file could be created in any directory, you may find it useful to create a directory just for this

purpose. The following example assumes that a directory called !Machine.Public has been created. (!Machine.Public is not part of the standard system.)

To see the message contained in a nondefault file:

1. Enter the following in a command window and press [Complete]:

   ```
   What.Message
   ```

2. Enter the name of the file at the prompt, replacing the name of the default file:

   ```
   What.Message (File => "!Machine.Public.Bulletin_Board");
   ```

3. Press [Promote].

Note that the ACL for the file should include Public=>R or Network_Public=>R. Like many other Environment commands, the What.Message procedure can be used from the operator's console command interpreter; however, the message can only be viewed, not edited.

If this command is run often, consider writing a simple "skin" that hard-codes the nondefault pathname, eliminating the need to type it each time.

Preparing Daily Messages in Advance

The What.Message command is useful for presenting users with general information when they log in. You may find at your installation that the message text needs to be updated infrequently—for example, once a week to inform users of the next scheduled backup. On the other hand, you may need to update the message text more frequently—for example, once a day to inform users of system schedules, company events, and the like.

To keep the message up to date, you can edit the message file on a daily basis. However, if you know in advance the messages you want to display on upcoming days, you can put those messages in a series of files corresponding to the dates of display and then create a procedure that displays the file corresponding to the current date.

To prepare a series of advance messages:

1. Create in !Users.Operator a series of files whose names follow this pattern:

   Msg_yy_mm_dd

   That is, each filename contains an alphabetic prefix Msg, followed by an underscore and then the date on which the message is to appear. The components of the date are in Ada format: year followed by month followed by day, separated by underscores. For example:

   ```
   Msg_92_08_28
   Msg_92_08_29
   Msg_92_08_30
   ```

   Put the desired message text in the appropriate files. (If there is no file for a particular day, no message will be displayed on that day.)

2. Create in !Local a procedure that displays the file for the current date. This procedure calls What.Message and supplies it with a filename that is constructed from the chosen prefix and the current date:

```
with What;
with Time_Utilities;

procedure Current_Message is
   package Tu renames Time_Utilities;
begin
   What.Message (File => "!Users.Operator.Msg_" &
                         Tu.Image (Tu.Get_Time,
                                   Date_Style => Tu.Ada,
                                   Contents   => Tu.Date_Only));
end Current_Message;
```

3. If you want to display the message unconditionally, add to !Machine.Current-.Commands a link to !Local.Current_Message, add to !Machine.System_Login a *with* to Current_Message, and then call Current_Message in System_Login. For example:

```
with Current_Message;
procedure System_Login is
begin
. . .
Current_Message;
. . .
end System_Login
```

4. If you want the system default login to display the message, add to !Machine-.Release.Current.Commands a link to !Local.Current_Message, add to !Machine-.Release.Current.Commands.Login a *with* to Current_Message, and then call Current_Message in the default login. For example:

```
with Current_Message;
procedure Login is
begin
. . .
Current_Message;
. . .
end Login
```

5. Any users who have customized login procedures should add to their home worlds a link to !Local.Current_Message and then modify their login procedures as shown in step 4.

10 Terminal Information

This chapter covers the following two major topics:

- *Hardware ports for terminal access:* Describes the different types of ports on the R1000 and tells where to look for more information about setting the configuration of these ports.
- *Terminal types and terminal-type objects:* Describes software that resides in the !Machine.Editor_Data library. Terminal types identify a set of input and output characteristics associated with a given terminal or terminal emulator. A *Terminal_Type* is the name of any terminal type that a user may enter at the `Enter Terminal Type:` prompt during login.

HARDWARE PORTS FOR TERMINAL ACCESS

The R1000 has two types of hardware ports for terminal access:

- Telnet ports
- RS232 asynchronous serial ports

Some Environment commands accept the Line parameter. This parameter is of type Port, and it identifies the physical *port number*. The Line parameter defaults to the number of the port through which your current session is logged in.

Table 10-1 summarizes the distribution of both types of ports on the various series of R1000s.

Table 10-1 Hardware Port Assignments for Series 200–400

Ports	Series 200	Series 300	Series 400	Notes
1	RS232	RS232	RS232	Used for operator's console to provide VT100-compatible, line-oriented Environment interface. Required to start machine.
2–15	Reserved	Reserved	Reserved	Reserved for internal hardware use.
16	RS232	RS232	RS232	Typically used for system manager's Environment terminal to provide a screen-oriented interface. On the Series 300, this port is labeled "External Modem." On the 400, it is labeled "Comm Port."
17–47	RS232	Unused	Unused	On the Series 200, the available range of ports (17–47) depends on how many serial-communications panels are installed. Typically, these ports are assigned to login terminals, printers, and other applications such as CDF interfaces.

Table 10-1 Hardware Port Assignments for Series 200–400 (continued)

Ports	Series 200	Series 300	Series 400	Notes
224–255	Telnet (upgrade for Model 10)	Telnet	Telnet	Available as an upgrade on Series 200 Model 10 machines; standard on Series 200 Model 20 and 40, Series 300, and Series 400 machines. The standard machine initialization for the current Environment release configures ports 235–249 as login ports.

Port Settings and Terminal Setup

Each terminal port is configured at installation. A port's configuration determines specific aspects of the communication between the Environment and the terminal or printer that is connected through the port. Each such aspect is controlled by an individual port-configuration setting.

To display the settings for a particular port (for example, port 17), execute the following command:

```
Terminal.Settings (17)
```

(If you omit the parameter from this command, the settings for your current port are displayed.)

A display similar to the following appears:

```
Terminal Settings for Port 17
-----------------------------
Terminal Type =                    RATIONAL
Input Baud Rate =                  BAUD_9600
Output Baud Rate =                 BAUD_9600
Parity =                           NONE
Stop_Bits =                         2
Char_Size =                        CHAR_8
Flow Control For Transmit Data =   NONE
Flow Control For Receive Data =    NONE
Disconnect_On_Disconnect =         FALSE
Disconnect_On_Logoff =             FALSE
Disconnect_On_Failed_Login =       FALSE
Log_Failed_Logins =                FALSE
Login_Disabled =                   FALSE
```

Certain Environment port settings correspond to—and therefore must match—specific hardware settings on the terminal itself. Consequently, if you change a port setting on the Environment, you must also verify that the corresponding setting on the terminal is the same (see your terminal users manual).

In Table 10-2, the "Value" column lists selected items from the above example display for port 17 when connected to a Rational terminal. The "Terminal Setup Option" column lists required terminal settings found in Chapter 3 of the *Rational Terminal User's Manual.* Note that they match, except for the parity setting, which is disabled on the R1000 port.

Table 10-2 Correspondences between Port Settings and Terminal Settings

Port Parameter	Value	Terminal Setup Parameter	Terminal Setup Option
Input baud rate	BAUD_9600	Main port speed	9600
Output baud rate	BAUD_9600	Main port speed	9600
Parity	NONE	Main/auxiliary parity	Mark
Flow control for transmit data	NONE	Generate Xon/Xoff	No
Flow control for receive date	NONE	Respond to Xon/Xoff	No

See Appendix E and the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, package Terminal, for a complete description of terminal ports, their settings, and related Environment commands.

Changing Terminal Port Settings

You can change the port settings by making changes to the machine-initialization procedure or by using Environment commands directly. Normally, you do not need to change the standard port settings unless you want to connect a terminal remotely or change the type of terminal connected to the R1000.

See Appendix E and the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, package Terminal, for complete information and procedures for changing terminal port settings.

The remainder of this chapter covers information about changing terminal types and key bindings.

TERMINAL TYPES AND TERMINAL-TYPE OBJECTS

The remainder of this chapter covers the software for terminal types and terminal-type objects.

A terminal type is defined in the Terminal_Types file and implemented in a set of key-binding objects and an output driver. A terminal type also can be associated with an autorecognition sequence that the R1000 uses to identify specific terminal types automatically during login.

In this chapter, whenever one of the following references appears, you can use any valid terminal type for *Terminal_Type:*

- *Terminal_Type*_Keys
- *Terminal_Type*_Key_Names
- *Terminal_Type*_Commands
- *Terminal_Type*_User_Commands

For example, if you are using the Pc101 terminal type, you use Pc101_Commands wherever *Terminal_Type*_Commands is specified.

OVERVIEW OF OBJECTS THAT DEFINE TERMINAL TYPES

Terminal-type objects define input and output characteristics of the terminals and workstations used with the Environment.

All terminal types are made up of at least the following three objects (where *Terminal_Type* stands for the name associated with each kind of terminal device that sends data to the Environment's editor):

- The *Terminal_Type*_Keys file in the !Machine.Editor_Data library
- The *Terminal_Type*_Key_Names Ada units in the !Machine.Editor_Data library
- The *Terminal_Type*_Commands Ada units in the !Machine.Editor_Data library

A terminal type can also include:

- An entry in the !Machine.Editor_Data.Terminal_Types file
- An entry in the !Machine.Editor_Data.Terminal_Recognition file

MANAGING THE TERMINAL_TYPES FILE

The Terminal_Types file contains definitions of terminal types used with RXI, RWI, and possibly other terminal-type implementations. (The Terminal_Types file does not define the Rational, Facit, or Vt100 terminal types. These types are defined internally in the Environment editor.) Definitions in the Terminal_Types file are used during both system elaboration and user login.

The Terminal_Types file typically is created when RXI or RWI is first installed. During subsequent installations of RXI, RWI, or other terminal emulators, new terminal-type definitions are added to the Terminal_Types file.

If RWI, RXI, or a similar terminal emulator is not installed on an R1000, then the !Machine.Editor_Data world probably does not contain a Terminal_Types file.

A typical Terminal_Types file looks like this:

```
Xhp46021a               XRTERM 60 80
Xhp         :xhp46021a  XRTERM 60 80
XApollo3                XRTERM 50 80
XAp         :XApollo3   XRTERM 50 80
XSun4                   XRTERM 79 80
Pc101                   Rational 66 80
```

Consider the entry:

```
XSun4                   XRTERM 79 80
```

The element XSun4 is a terminal-type name associated with RXI. This terminal-type name can be entered by the user in response to the `Enter Terminal Type:` prompt. The terminal-type name is also associated with a set of key-binding objects located in the !Machine.Editor_Data world and is required when making custom modifications.

Terminal-type definitions have the following syntax:

`Terminal_Type` `[:`*`AliasedType`*`] [.`*`CustomKeymap`*`] [`*`Output`*`] [`*`Lines`* `[`*`Cols`*`]]`

The following paragraphs give a detailed description of each element in the terminal-type definition.

.Custom Keymap (Optional)

.Custom Keymap is an optional element that names the prefix for a *Terminal_Type*_Commands custom key-binding procedure and any accompanying *Terminal_Type*_Macros custom macros that exist in the user's home library (or other library indicated by the user's Key_Directory session switch). Note that this element is needed only if the prefix for the custom *Terminal_Type*_Commands procedure differs from the prefix for the systemwide *Terminal_Type*_Commands procedure: by default, the Environment assumes that any relevant custom key bindings will have the same name as the systemwide key bindings.

See "Defining Nondefault Names for Custom Key Bindings" in the next subsection for an example application.

:Aliased Type (Optional)

:Aliased Type is an optional element that names another terminal type defined earlier in the file. If this optional element is included, then the new terminal type is an alias for the key-binding portion of the previously defined terminal type and is associated with the same set of key-binding objects. No new key-binding objects are defined for the new terminal type. An alternate output driver can be specified, as well as new display dimensions.

If the same output driver and display dimensions are used, then the new terminal type is a true alias of the previously defined terminal type. One reason this might be done is to provide a shorter name for a long or awkward terminal-type name. A shorter name is easier to enter at the `Enter Terminal Type:` prompt during login.

Adding an Alias

You can add entries to the Terminal_Types file that use the *:Aliased Type* component to alias a terminal type that already exists on the system. A new terminal type, or *alias*, that is defined with an aliased type inherits the key bindings of the aliased type. No new key-binding units are required.

An alias should be a unique terminal-type name.

Aliases can provide shorter terminal names and alternative window sizes, making it easier for the user to specify terminal types.

- Creating aliases for shorter names:

  Suppose the Terminal_Types file contains the following terminal-type definition:

  ```
  XC2BRLS1948 XRTERM 66 80
  ```

  XC2BRLS1948 is cumbersome to enter. If you add the alias:

  ```
  XC :XC2BRLS1948 XRTERM 66 80
  ```

  to the Terminal_Types file following the XC2BRLS1948 entry, then users can specify the XC terminal-type alias instead of XC2BRLS1948.

- Creating aliases to achieve alternative window sizes:

  Suppose a workstation is associated with the following entry in the Terminal_Types file:

  ```
  XNorm XRTERM 66 80
  ```

This entry specifies a window dimension of 66 lines by 80 columns.

If you add the alias:

```
XWide :XNorm XRTERM 66 160
```

to the Terminal_Types file following the XNorm entry, users can specify the XWide entry at the `Enter Terminal Type:` prompt during login, instead of:

```
XNorm 66 160
```

To add an alias:

1. Traverse to the !Machine.Editor_Data.Terminal_Types file.
2. Open the file for editing by pressing [Edit].
3. Make sure that the aliased terminal-type definition does not already exist in the file.
4. Insert the new terminal-type definition containing the alias at the end of the file. For example:

   ```
   rwi :Pc101 Rational 66 80
   ```

5. Press [Promote] to update the file.
6. Either wait until the R1000 reboots or execute the command:

   ```
   Refresh_Terminal_Information;
   ```

Defining Nondefault Names for Custom Key Bindings

The *.Custom Keymap* and the *:Aliased Type* components can be combined in a terminal-type definition to create a related set of custom keymaps.

By default, the Environment looks for a custom key-binding procedure in the user's home library during login, where the custom key-binding procedure has the same prefix name as the systemwide key-binding procedure located in !Machine-.Editor_Data. (The prefix name is, by default, the same as the terminal-type name.)

If a *.Custom Keymap* entry is included as part of a terminal-type definition in the Terminal_Types file, then a custom key binding with a nondefault prefix can be specified.

For example, if a system manager adds the following three lines to the Terminal-_Types file:

```
Keyset1 :XSun4 .Set1 XRTERM 66 80
Keyset2 :XSun4 .Set2 XRTERM 66 80
Keyset3 :XSun4 .Set3 XRTERM 66 80
```

then three aliases are defined for the XSun4 terminal type: Keyset1, Keyset2, and Keyset3. These three aliases are associated with the custom key-binding procedures: Set1_Commands}, Set2_Commands, and Set3_Commands, respectively.

Now if a user has access to these three key-binding procedures (either in the user's home library or another directory indicated by the Key_Directory session switch), then the user can enter: `Keyset1`, `Keyset2`, or `Keyset3` at the `Enter Terminal Type:` prompt during login to specify one of these custom keymaps.

To define a nondefault name for a custom key binding, use the same set of steps for adding an alias, except add a *.Custom Keymap* entry after the *:Aliased Type* entry (see "Adding an Alias," above).

Output

Output specifies which *output driver* manages data sent from the R1000 to the terminal device.

The Rational Environment supports the XRTERM, Rational, Facit, and Vt100 output drivers:

- The XRTERM output driver manages data sent from an R1000 to a workstation running RXI. Note that although XRTERM is part of the Rational Environment, it is not available for use until RXI has been installed on the R1000.
- The Rational output driver manages data sent from an R1000 to Rational terminals and to PCs running RWI.
- The Facit controls output data sent from an R1000 to a Facit terminal.
- The Vt100 controls output data sent from an R1000 to a VT100-compatible terminal.

Lines and Cols (Optional)

Lines specifies the number of character lines supported by the terminal. If this component is omitted, the default is 24 lines.

Cols specifies the number of character columns supported by the terminal. If this component is omitted, the default value is 80. Note that if you specify the number of columns, you must also specify the number of lines.

When logging in, the user can override the values defined in the Terminal_Types file by entering the terminal type and window dimensions at the `Enter Terminal Type:` prompt (see "Logging In without Autorecognition" and "Overriding Autorecognition," below). This allows matching an arbitrary RXI or RWI window size to the Environment's window manager.

TERMINAL RECOGNITION

The installation of RXI or RWI creates a Terminal_Recognition file if this file does not exist. An entry for the new workstation or PC terminal type is added automatically. This entry corresponds to the terminal type defined in the Terminal_Types file.

The Rational, Facit, and Vt100 terminal types are not defined in the Terminal_Recognition) file. The standard Environment is coded internally to recognize these three terminals. You can enter new entries based on these terminal types, however, to provide an *:Aliased Type* or *.Custom Keymap.*

A typical Terminal_Recognition file looks something like this:

```
XDecUS [ESC][?1;34c
XApollo3 [ESC][?1;82c
XNCD [ESC][?1;42c
XNews3 [ESC][?1;50c
XNews4 [ESC][?1;58c
XRTUS [ESC][?1;26c
```

```
XSun3 [ESC][?1;10c
Pc101 [ESC]pc101c
XSun4 [ESC][?1;18c
Xhp46021a [ESC][?1;66c
```

Each entry has the following syntax:

Terminal_Type Recognition Sequence

The recognition sequence is limited to 32 characters.

Entries in the Terminal_Recognition file are read sequentially. If an entry later in the file specifies the identical recognition sequence as an earlier entry, then the entry appearing later in the file will override the earlier one. Consequently, when a terminal corresponding to the first entry attempts to log in, the R1000 will recognize it as the subsequent terminal type, which may be incorrect.

If the terminal-recognition entry is missing for a particular terminal type, users will have to specify the terminal type at the the `Enter Terminal Type:` prompt before they log in (see "Logging In without Autorecognition" and "Overriding Autorecognition," below).

You can add a new terminal-recognition entry to the Terminal_Recognition file if one is missing or if you create a new terminal type. You must obtain the ANSI-standard recognition sequence from the manufacturer of the terminal device. (See the caution at the beginning of the "Understanding the Terminal-Type and Key-Binding Objects" section.)

How Autorecognition Works

During login the Environment configures the terminal device's terminal type by performing the following series of steps.

1. The Environment sends the ANSI terminal-device identification escape sequence—ESC[0c—to the terminal device. The terminal device responds by sending the unique ANSI-standard recognition escape sequence (see above).
2. The Environment attempts to match this recognition sequence with the recognition sequences of the predefined terminal types: Rational, Facit, or Vt100.
3. If the Environment does not find a match, then a table, created at boot time from the !Machine.Editor_Data.Terminal_Recognition file, is searched sequentially for a matching recognition sequence.

   As with the Terminal_Types file, the Terminal_Recognition file is not part of the standard Environment. The Terminal_Recognition file is created automatically, and recognition sequences for the terminal types associated with a particular RXI or RWI implementation are added when these products are installed.
4. If the recognition sequence from the terminal device is not found, the Environment prompts the user to enter the terminal type. (See "Logging In without Autorecognition," below, for more information.)

Logging In without Autorecognition

If the R1000 cannot automatically determine the terminal type, you are prompted for it when logging in; the terminal type last used appears in parentheses. The following example shows that a Facit terminal was last connected to the port:

```
Enter terminal type (FACIT):
```

To log in, enter the terminal type—for example, xsun4 (case is insignificant):

```
Enter terminal type (FACIT): xsun4 [Return]
```

Overriding Autorecognition

You can override autorecognition and force the terminal-type prompt to appear by entering an equals sign (=) at the username prompt and pressing [Return]:

```
Enter user name: = [Return]
```

This is useful when you want to override the default window dimensions defined in the !Machine.Editor_Data.Terminal_Types file or use a terminal type other than the one specified through autorecognition.

You can enter your preferred dimensions after the terminal type. The following example specifies 76 lines by 150 columns for the xsun4 terminal type:

```
Enter terminal type (FACIT): xsun4 76 150 [Return]
```

After you enter the terminal-type information, the username prompt returns and you can continue logging in as usual.

UNDERSTANDING THE TERMINAL-TYPE AND KEY-BINDING OBJECTS

Caution: *Creating a new systemwide terminal type and key bindings is beyond the scope of this document. The systemwide terminal-type objects are developed with tools that ensure consistency and accuracy. The systemwide Terminal_Type_Keys, Terminal_Type_Key_Names, and Terminal_Type_Commands key-binding objects should not be edited. If you need a new terminal type, contact Rational.*

This section identifies each of the terminal-type objects, describes what they do, and identifies relationships between them. Terminal-type objects are discussed under the following subsections:

- "Key Naming and Character Recognition"
- "Environment Key Bindings"
- "Installing Systemwide Key Bindings"
- "Installing and Customizing Key Bindings for a Terminal Type"
- "Creating Key Directories Associated with Specific Sessions"

Key Naming and Character Recognition

For the Environment to recognize which keys are pressed on a given terminal's keyboard, each terminal type includes a list of character definitions called the *Terminal_Type*_Keys file. As a user convenience, an enumeration of keynames is included in package *Terminal_Type*_Key_Names.

Package *Terminal_Type*_Key_Names

Package *Terminal_Type*_Key_Names contains an enumeration type whose literals form the names of all R1000-supported keys for the terminal device's keyboard (some keys may not transmit and some keys may perform functions that are

unsupported by the R1000, so it is not always true that every physical key on a keyboard will have a name used in the Environment).

The *Terminal_Type*_Key_Names unit is an end-user convenience, providing human-readable keynames instead of the symbols in the *Terminal_Type*_Keys file.

Only the Vt100_Key_Names, Rational_Key_Names, and Facit_Key_Names packages are included as part of the standard Environment.

The *Terminal_Type*_Keys File

The Terminal_Type_Keys text file specifies to the Environment what characters to receive and what keystrokes to infer based on those received characters. Two maps are created from this file: the first is a two-way map between key names and the Environment's internal key-code values; the second is a one-way map from ASCII character sequences to internal key codes.

Only the Vt100_Keys, Rational_Keys, and Facit_Keys files are included as part of the standard Environment.

Environment Key Bindings

The actual Environment key bindings are implemented as a default systemwide standard that can be supplemented optionally by user customizations.

Procedure *Terminal_Type*_Commands

Each terminal type is associated with a *Terminal_Type*_Commands key-binding procedure located in !Machine.Editor_Data. The body of *Terminal_Type*_Commands is a long case statement divided into three sections: Interrupt, Prompt, and Execute. Nested case statements within each section bind keystroke sequences to R1000 system commands. This procedure is commonly called the *keymap* or the *key bindings* for a terminal type.

Only the Vt100_Commands, Rational_Commands, and Facit_Commands procedures are included as part of the standard Environment. Others, such as XSun4-_Commands, are included when terminal-emulator products such as RXI or RWI are installed.

Read access to these units can help users who are trying to create custom key bindings. The systemwide key bindings, however, should not be edited.

The *Terminal_Type*_User_Commands File

The *Terminal_Type*_User_Commands files located in !Machine.Editor_Data provide a basic framework that allows a user to customize key bindings for the user's session. A typical file looks like this:

```
with Terminal_Type_Key_Names;
procedure Terminal_Type_Commands is
    use Terminal_Type_Key_Names;
    type Intent is (Prompt, Execute, Interrupt);
    Action : Intent;
    Key_1, Key_2, Key_3, Key_4, Key_5, Key_6  : Key_Names;
```

```
begin
    case Action is
        when Prompt =>
            case Key_1 is
                when others => null;
            end case;
        when Execute =>
            case Key_1 is
                when others => null;
            end case;
        when Interrupt =>
            case Key_2 is
                when others => null;
            end case;
    end case;
end Terminal_Type_Commands;
```

This file has the same overall structure as the *Terminal_Type*_Commands procedure described above in the "Procedure *Terminal_Type*_Commands" subsection. The *Terminal_Type*_User_Commands file is a text file, however, not an Ada unit. Users parse this template into a custom *Terminal_Type*_Commands Ada unit within their home library. Refer to the "Installing and Customizing Key Bindings for a Terminal Type" subsection, below, for more information on creating a custom key binding from this template.

Installing Systemwide Key Bindings

This subsection covers the systemwide key bindings only: procedures for user's custom key bindings appear in the next subsection.

Caution: *Use care when modifying objects in !Machine.Editor_Data so that the systemwide binding mechanism is not corrupted. Otherwise, users may be prevented from logging in.*

Each *Terminal_Type*_Key_Names and *Terminal_Type*_Commands unit should be in at least the installed state. The Environment uses the semantic information found in the installed keymaps. The semantic information is still present if the *Terminal_Type*_Commands unit is promoted to the coded state. However, the machine-code segments generated by this extra step are never used and they occupy disk space.

If the Ada units are not in the installed state, follow these steps:

1. Select *Terminal_Type*_Key_Names first; promote it to the installed state by pressing [Promote].
2. Select *Terminal_Type*_Commands'Spec second; promote it to the installed state.
3. Select *Terminal_Type*_Commands'Body third; promote it to the installed state.
4. Immediately after *Terminal_Type*_Commands is promoted to the installed state, open a command window under the !Machine.Editor_Data directory image and execute the command:

   ```
   Refresh_Terminal_Information;
   ```

This runs for about one minute for each *Terminal_Type*_Commands unit in the directory. (Each Environment terminal type has a *Terminal_Type*_Commands unit associated with it.)

When the Environment boots or when the Refresh_Terminal_Information command is issued, if some *Terminal_Type*_Commands unit is not in at least the installed state, the Environment attempts to promote it to that state. If the promotion succeeds, the Environment continues: it reads the installed unit and creates an internal optimized version for its immediate use. If the promotion fails, any terminal types that reference this *Terminal_Type*_Commands unit as their keymap are not available. (If there is a syntax or semantic error in the *Terminal_Type*-_Commands unit, the unit will not promote. This is why you promote the units before executing Refresh_Terminal_Information, just to be sure.)

Installing and Customizing Key Bindings for a Terminal Type

A user can create custom key bindings that supplement or override certain system-wide key bindings for the user's session. The steps for doing this are organized into two general tasks:

- "Preparing a Custom Key Binding Skeleton in a User's Home Library": First the user copies the !Machine.Editor_Data.*Terminal_Type*_User_Commands file to his or her home directory, parses the file into a local *Terminal_Type*_Commands Ada unit, sets up some links, and installs the new unit as a custom key-binding skeleton. This task needs to be done only once by the user.
- "Adding Custom Key Bindings to the Skeleton": After the skeleton is properly set up, the user adds or deletes custom key bindings when needed by: demoting the custom key-binding unit, using the editor to update the bindings, promoting the revised custom key-binding unit, and then logging out and back in again to activate the new bindings.

Both of these tasks are detailed in the following two subsections.

Preparing a Custom Key-Binding Skeleton in a User's Home Library

To parse the *Terminal_Type*_User_Commands file into an Ada unit that customizes the key bindings defined by the *Terminal_Type*_Commands procedure, follow these steps:

1. Traverse to your home library.
2. To ensure that you have a link to the Ada unit *Terminal_Type*_Key_Names, open a command window and run:

```
Links.Add
    (Source => "!Machine.Editor_Data.Terminal_Type_Key_Names");
```

3. To make your own skeleton key binding of the *Terminal_Type*_Commands unit, run the following command from the command window:

```
Compilation.Parse
    (File_Name =>
          "!Machine.Editor_Data.Terminal_Type_User_Commands",
     Directory => "$");
```

This parses the *Terminal_Type*_User_Commands file into your home library as the custom Ada unit *Terminal_Type*_Commands.

4. Install the new *Terminal_Type*_Commands procedure by pressing [Promote] once. If the unit does not install, resolve semantic errors by checking the above steps and making sure that the *Terminal_Type*_User_Commands file is correct.

Adding Custom Key Bindings to the Skeleton

Before following these steps, make sure that you have completed "Preparing a Custom Key-Binding Skeleton in a User's Home Library," described above. The following steps show how to bind an Environment command to a key. Assume, for example, that your system supports the Mail package, that your keyboard includes the keys: [Control] [Meta] [~], and that you want to bind the Mail.Edit command to that key combination.

1. Go to your home library if you are not already there.
2. Decide what command you want to bind. For this example, use Mail.Edit.
3. Check for any existing key bindings for the command by executing:

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

   The first line of the help window created by What.Does contains the pathname of the command:

   ```
   !Commands.Mail.Edit
   ```

   Note the pathname to the package (!Commands.Mail) for use in step 6.

   If there are existing key bindings for the command, they are listed beginning on the first line immediately following the pathname. If key bindings exist, you may decide not to make a new binding and use the existing one instead.
4. Determine what key or key sequence you want to bind the command to. For example, you might want to bind Mail.Edit to [Control] [Meta] [~]. Check to see if the systemwide key bindings already contain a key binding for this key by pressing [Prompt For] (the Editor.Key.Prompt command) to get the prompt:

   ```
   Prompt For:
   ```
5. Now press the prospective key sequence [Control] [Meta] [~].

   If there are no key bindings, the message window shows:

   ```
   Prompt For: CM_GRAVE, undefined key.
   ```

   (Note that the key name, CM_GRAVE, is displayed. You use this name when you code your custom key binding.)

   If the key is bound in the systemwide key bindings, then a command window appears that contains the command.

   This step is important, because any key bindings you make will override the respective systemwide key bindings: for example, if the desired key is the only key binding for the logical key [Promote], and you bind something new to that key, then [Promote] is no longer available. If you need to have the command bound to a key, either move the preexisting key binding to another key or choose another key for your new key binding.
6. To ensure that your home library has a link to package Mail, create a command window and run the Links.Add command with the pathname for the command (!Commands.Mail):

   ```
   Links.Add (Source => "!Commands.Mail");
   ```
7. Select the body of the Ada unit *Terminal_Type*_Commands. (The body is the second Ada item in the library directory with the name *Terminal_Type*_Commands. The first item with that name is the specification.)

8. Edit the body by pressing [Edit].
9. You need to include a *with* clause at the top of your *Terminal_Type*_Commands unit for the package containing the command. For the Mail.Edit example, the package name is Mail.

   So you add the `with Mail;` clause near the top of the unit:

   ```
   with Terminal_Type_Key_Names;
   with Mail; -- you add this line
   ```

   (See the *Reference Manual for the Ada Programming Language* for a detailed explanation of the *with* clause.)
10. Now search through *Terminal_Type*_Commands to see if there is already a key binding for each of the desired keys, in this case CM_GRAVE. If there is a key binding, make the same considerations you made in step 2, deciding either to find another key or to change the key binding.
11. The intended action is Execute, so modify that portion of the code to look like this:

    ```
    when Execute =>
        case Key1 is
            when Cm_Grave =>
                Mail.Edit
            when others => null;
        end case;
    ```

    Note: *This key may not be available on certain keyboards.*
12. Select the Mail.Edit line and press [Complete]. This formats and completes the command, so that it looks like this:

    ```
    when Execute =>
        case Key1 is
            when Cm_Grave =>
                Mail.Edit (Mailbox => "MAIN", For_User => "");
            when others => null;
        end case;
    ```

    You can now modify the parameters to suit your needs.
13. Install the Ada unit by pressing [Promote]. You should see the message:

    ```
    Terminal_Type_COMMANDS'Body changed to INSTALLED
    ```
14. Now log out and log back in. Your modified key binding should be active. You can try the [Control] [Meta] [~] key sequence to be sure.

Note: *If the custom key-binding procedures are lengthy, user login time increases. On a heavily used system, this may result in several minutes delay.*

Creating Key Directories Associated with Specific Sessions

A user can create specialized key-binding libraries (key directories) within the user's home world. Each key-binding library can be associated with a specific session through the Key_Directory session switch. Each key-binding library also contains a different version of an installed key-binding unit and a library switch file with the Create_Internal_Links switch set to False. (When the Create_Internal-_Links library switch is set to False for one or more directories contained within a world, those directories can contain installed Ada units having the same name as other Ada units installed anywhere else in that world. For more information, see the Session and Job Management (SJM) book of the *Rational Environment Reference Manual.*)

The general steps for creating a key-binding library and associating this library with a specific session are:

1. Create a specific session.
2. Open that session's switch file for editing.
3. Edit the Key_Directory session switch, giving it the unique name of a library that you will create.
4. Promote the session switch file.
5. Create the library, using the name from step 3.
6. Traverse to the library, and create a Library_Switch file by executing the Switches.Edit command.
7. Edit the Create_Internal_Links switch so that it is set to False.
8. Promote the Library_Switches file.
9. Follow the instructions described in the "Installing and Customizing Key Bindings for a Terminal Type" subsection, above, except install the key-binding unit in this new library rather than the home world.

Now when the user logs in specifying the session associated with the new key-binding library, the new key bindings will take effect.

11 Overview of Series 200 System

This chapter provides an overview of the Rational R1000 Series 200 Development System. This development system consists of the R1000 Series 200 processor, the Rational Environment, and supporting peripherals.

The R1000 Series 200 System (200S) hardware is configured according to customer specifications. The standard base configurations are described in the first section of this chapter.

Refer to illustrations in this chapter to help locate and identify system components.

SERIES 200 CONFIGURATIONS

The R1000 Series 200 processor supports the Rational Environment. It can be configured to support up to four disk drives, 32 communication ports, and two tape drives (one 9-track and one 8-mm) per CPU bay. In all standard configurations, the front and back cabinet doors can be locked to protect internal hardware from damage and to protect users from hazards such as electric shock. The following are the standard Series 200 configurations:

- Model 10 (see Figures 11-1, 11-2, and 11-3)
  - One bay for the CPU and peripherals
- Model 20 (see Figures 11-4, 11-5, and 11-6)
  - One CPU bay
  - One peripheral bay
- Model 20B (see Figures 11-4, 11-7, and 11-8)
  - One CPU bay
  - One peripheral bay
- Model 40 (see Figure 11-9)
  - Two CPU bays
  - One peripheral bay
  - One optional expansion bay

    This optional expansion bay is added on the left of the three-bay Model 40 when any of the following configuration requirements exist:
    - A second 9-track tape drive is installed on the system (see Chapter 15)
    - A fourth 10-inch disk drive is installed for either CPU (see Chapter 16)
    - A second communications panel is installed for CPU-0

Model 10

The Model 10 is a one-bay standalone unit containing the R1000 processor boards, one or two tape drives, one to four 8-inch hard-disk drives, and a 16-port communications panel to which you typically can connect one printer and up to 15 user terminals or communication servers. The Model 10 supports the Rational Environment. See Figure 11-1 for an illustration of the front of the Model 10 cabinet. See Figure 11-2 for an illustration of the Model 10 with the front door open, and see Figure 11-3 for an illustration of the Model 10 with the rear door open. The Model 10 RS232C communication ports are installed in the communications panel, which is mounted in the bottom of the cabinet. The printer, user terminals, and communication servers are connected in these ports. See Figure 11-11 for a detailed illustration of the communications panel. The Model 10 control panel is located along the top of the front door. See Figure 11-12 for a detailed illustration of the control panel.

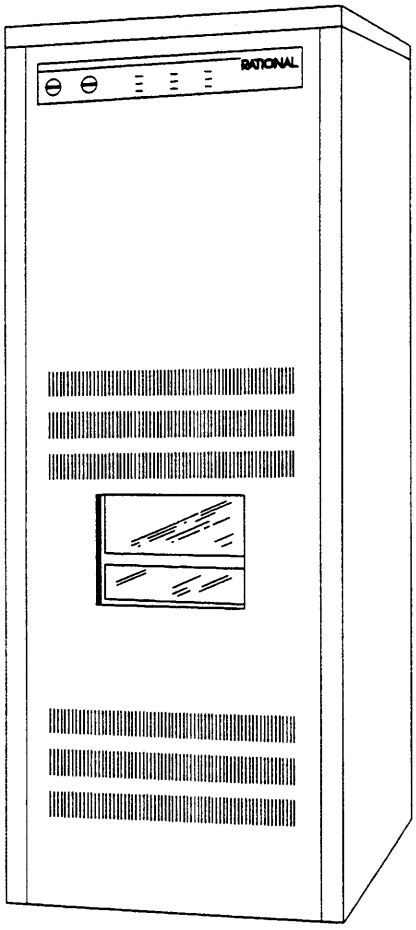

Figure 11-1 Series 200, Model 10, Cabinet Front

Model 10, Front Interior View

When you open the front door of the CPU bay of the Model 10 (see Figure 11-2), notice the locations of:

- The R1000 processor boards
- The connectors for the operator's console and the optional Ethernet network
- The disk-drive enclosure, containing a minimum of one and a maximum of four 8-inch disk drives (see Chapter 16)
- The 9-track tape drive
- The communications panel containing 16 RS232C communications ports

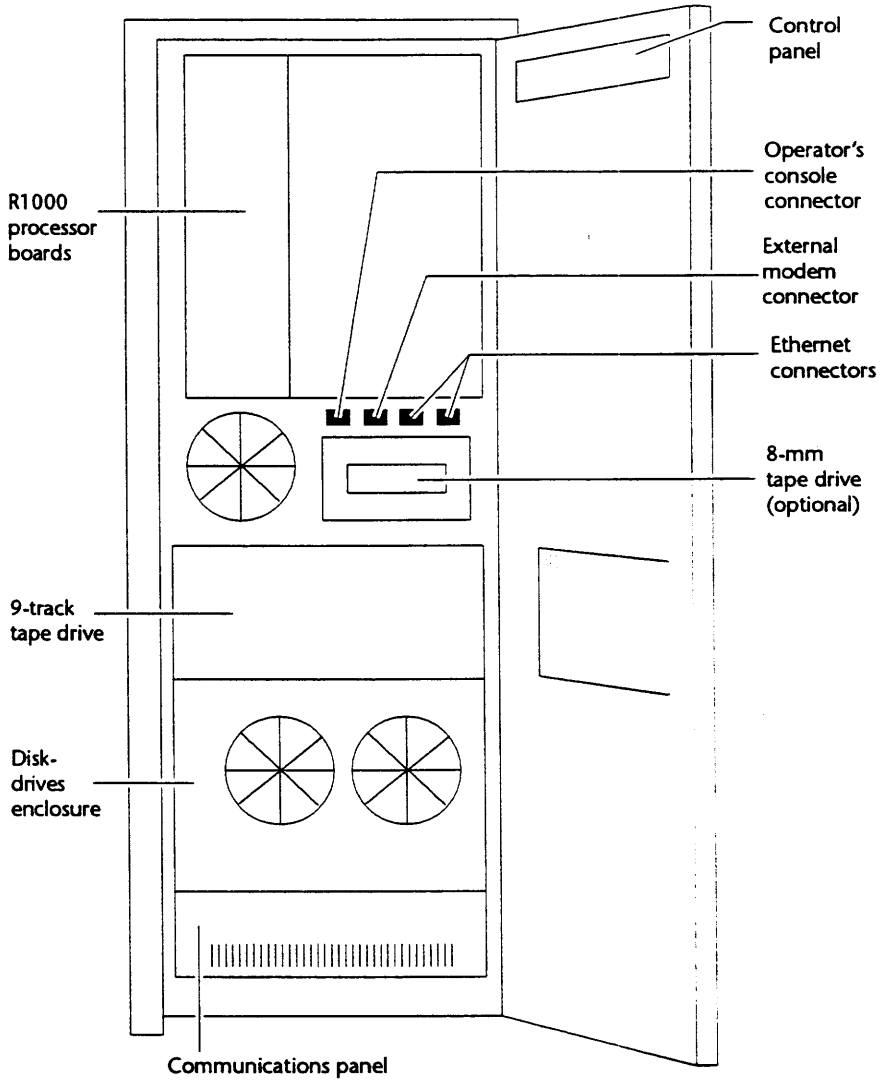

Figure 11-2 Series 200, Model 10, Front Interior View

Model 10, Rear Interior View

When you open the rear door of the Model 10 bay (see Figure 11-3), notice the locations of:

- The system and disk power supplies.
- The AC power-distribution unit (PDU) with the main circuit breaker (CB1).
- The power control and modem (PCM), which contains the telephone jack and the remote diagnostic modem through which the R1000 reports system failures to Rational's Response Center. If allowed and enabled, support personnel at Rational may use this remote diagnostic modem to conduct remote diagnostic sessions. (For details about the Response Center, see Appendix D, "Rational Customer Support Services.")

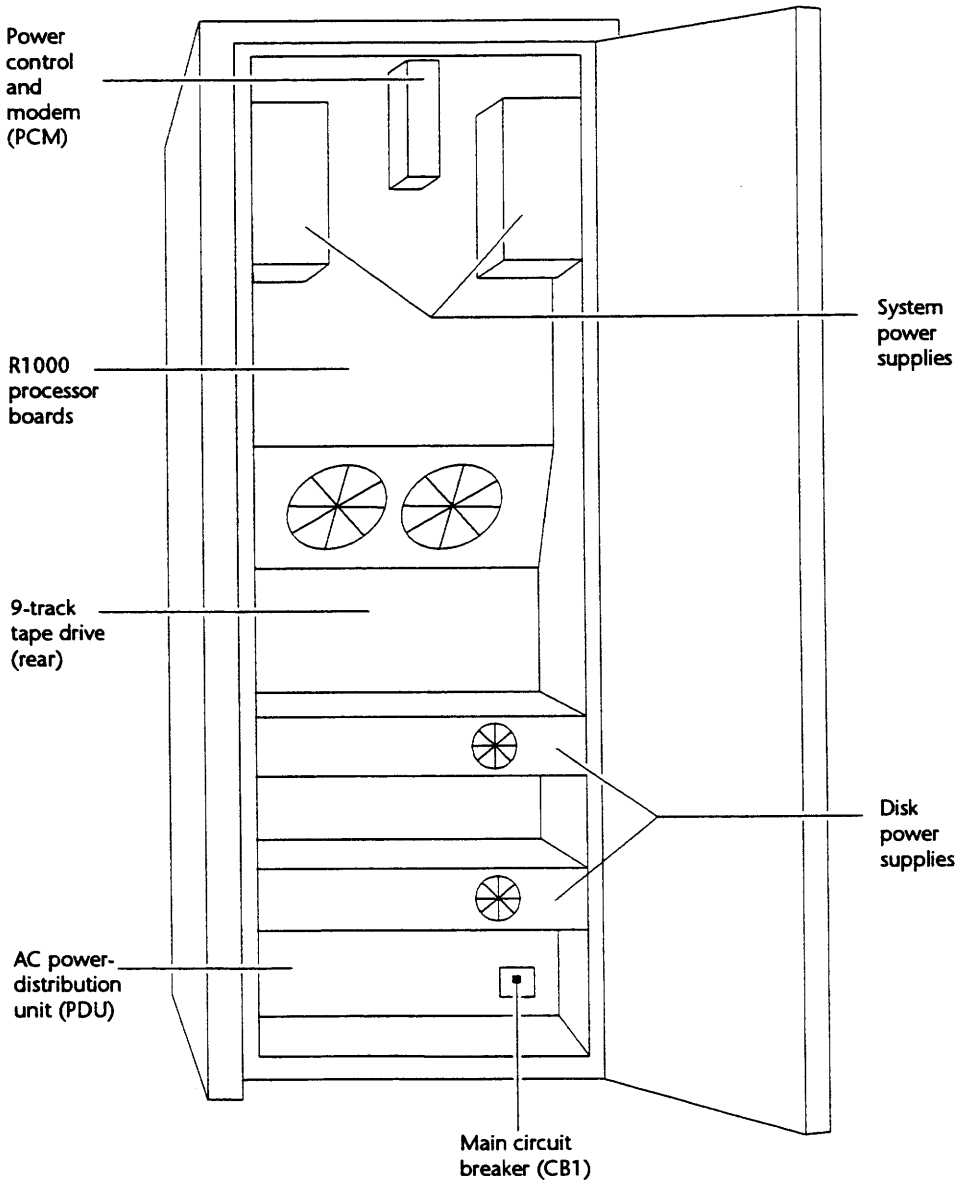

Figure 11-3 Series 200, Model 10, Rear Interior View

Model 20

The Model 20 is a two-bay standalone unit containing the R1000 processor boards, one or two tape drives, one to four 10-inch hard-disk drives, and one or two 16-port communication panels to which you typically can connect one or more printers and up to 15 (or 31) user terminals or communication servers. The Model 20 supports the Rational Environment. See Figure 11-4 for an illustration of the front of the Model 20 cabinet. See Figure 11-5 for an illustration of the Model 20 with the front doors open. See Figure 11-6 for an illustration of the Model 20 with the rear doors open. The Model 20's RS232C communications ports are installed in one or two communications panels, which are mounted in the bottom of one or both bays. See Figure 11-11 for a detailed illustration of a communications panel. The Model 20's control panel is located along the top of the CPU bay's front door. See Figure 11-12 for a detailed illustration of the CPU bay's control panel.

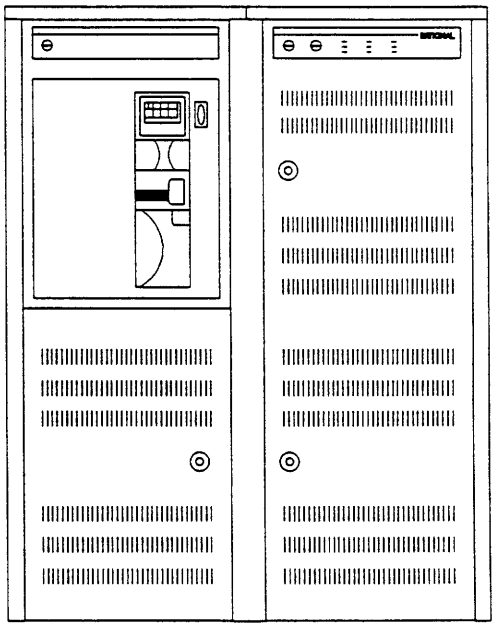

Figure 11-4 Series 200, Models 20 and 20B, Cabinet Front

Model 20, Front Interior View

When you open the front doors of the Model 20 bays (see Figure 11-5), notice the locations of:

- The R1000 processor boards
- The connectors for the operator's console, the optional external modem, and the optional Ethernet network
- The 10-inch disk drives: one or two in the CPU bay and one or two in the peripheral bay (see Chapter 16)
- The 9-track tape drive, mounted in the peripheral bay
- One optional 8-mm tape drive, mounted in the CPU bay
- One (or two) communications panels containing 16 (or 32) RS232C communications ports

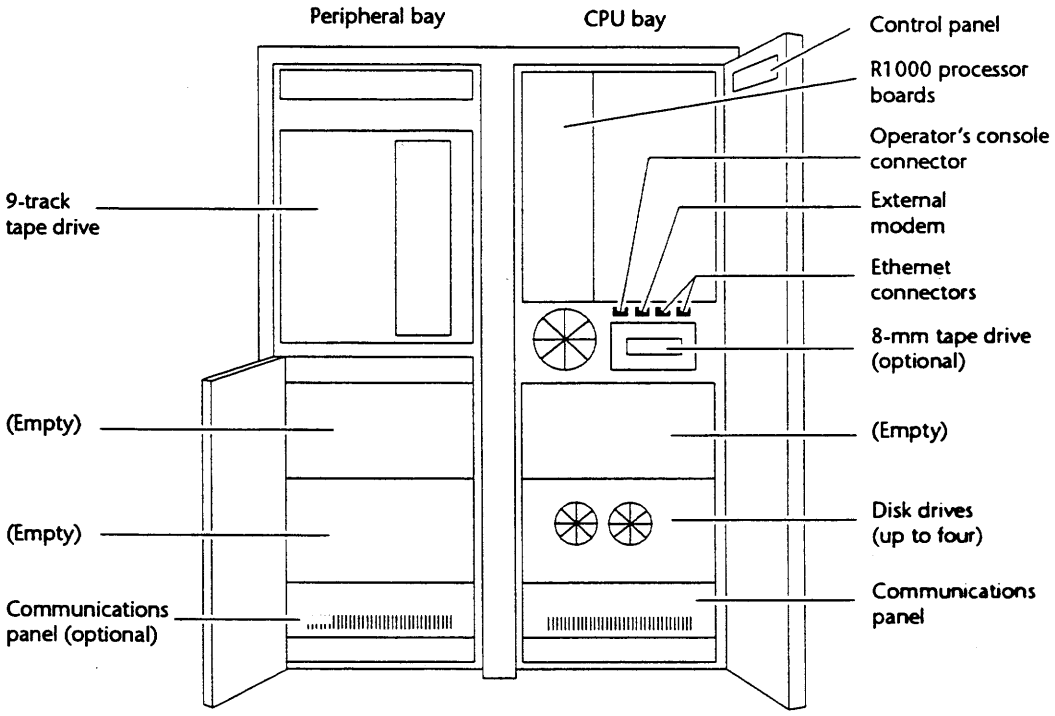

Figure 11-5 Series 200, Model 20, Front Interior View

Model 20, Rear Interior View

When you open the rear doors of the Model 20 bays (see Figure 11-6), notice the locations of:

- The system and disk power supplies.
- The AC power-distribution unit (PDU) with the main circuit breaker (CB1).
- The power control and modem (PCM), which contains the telephone jack and the remote diagnostic modem through which the R1000 reports system failures to Rational's Response Center. Support personnel at Rational may use this remote diagnostic modem to conduct remote diagnostic sessions. (For details about the Response Center, see Appendix D, "Rational Customer-Support Services.")

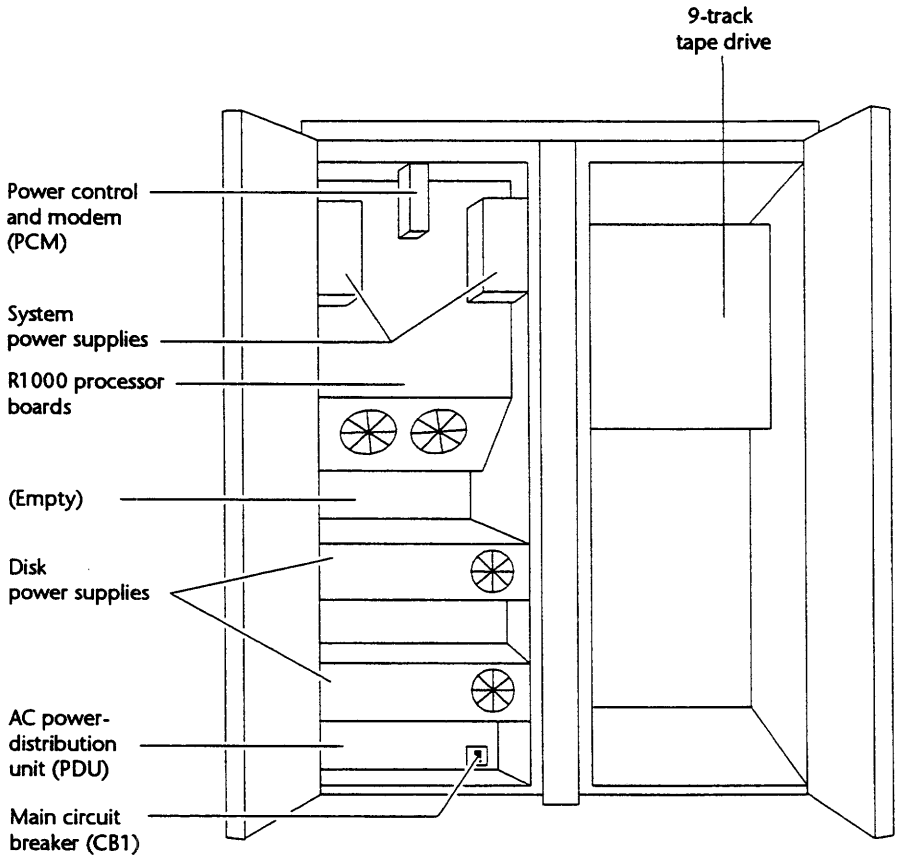

Figure 11-6 Series 200, Model 20, Rear Interior View

Model 20B

The Model 20B is a two-bay standalone unit containing the R1000 processor boards, one or two tape drives, one to four 8-inch hard-disk drives, and one or two 16-port control panels to which you typically can connect one printer and up to 15 (or 31) user terminals or communication servers. The Model 20B supports the Rational Environment. The Model 20B is the same as the Model 20 except that the Model 20B contains 8-inch disk drives, all four of which are mounted in the CPU bay. See Figure 11-4 for an illustration of the front of the Model 20 and Model 20B cabinets. See Figure 11-7 for an illustration of the Model 20B with the front doors open. See Figure 11-8 for an illustration of the Model 20B with the rear doors open. The Model 20B's RS232C communication ports (where user terminals are connected) are installed in one or two communications panels. See Figure 11-11 for a detailed illustration of a communications panel. The Model 20B's control panel is located along the top of the front door. See Figure 11-12 for a detailed illustration of the CPU bay's control panel.

Model 20B, Front Interior View

When you open the front doors of the Model 20B bays (see Figure 11-7), notice the locations of:

- The R1000 processor
- The connectors for the operator's console, the optional external modem, and the optional Ethernet network
- The disk drives, one to four, mounted in the CPU bay (see Chapter 16)
- The 9-track tape drive, mounted in the peripheral bay
- One optional 8-mm tape drive, mounted in the CPU bay
- One (or two) communications panels containing 16 (or 32) RS232C communications ports

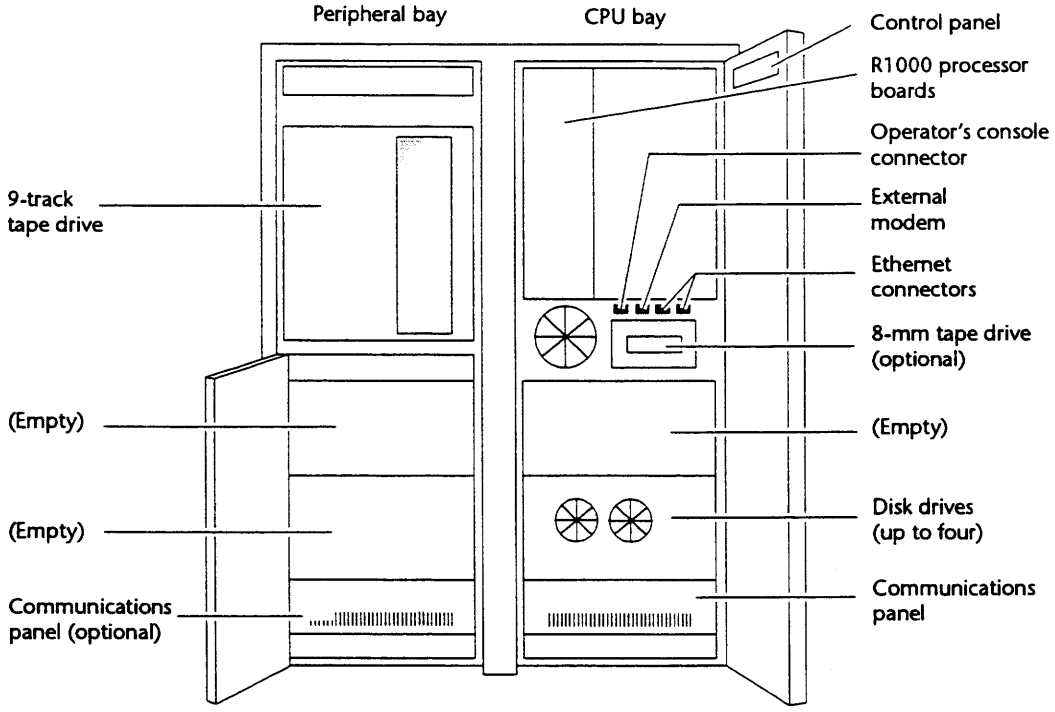

Figure 11-7 Series 200, Model 20B, Front Interior View

Model 20B, Rear Interior View

When you open the rear doors of the Model 20B bays (see Figure 11-8), notice the locations of:

- The system and disk power supplies.
- The AC power-distribution unit (PDU) with the main circuit breaker (CB1).
- The power control and modem (PCM), which contains the telephone jack and the remote diagnostic modem through which the R1000 reports system failures to Rational's Response Center. When allowed and enabled, support personnel at Rational may use this remote diagnostic modem to conduct remote diagnostic sessions. (For details about the Response Center, see Appendix D, "Rational Customer-Support Services.")

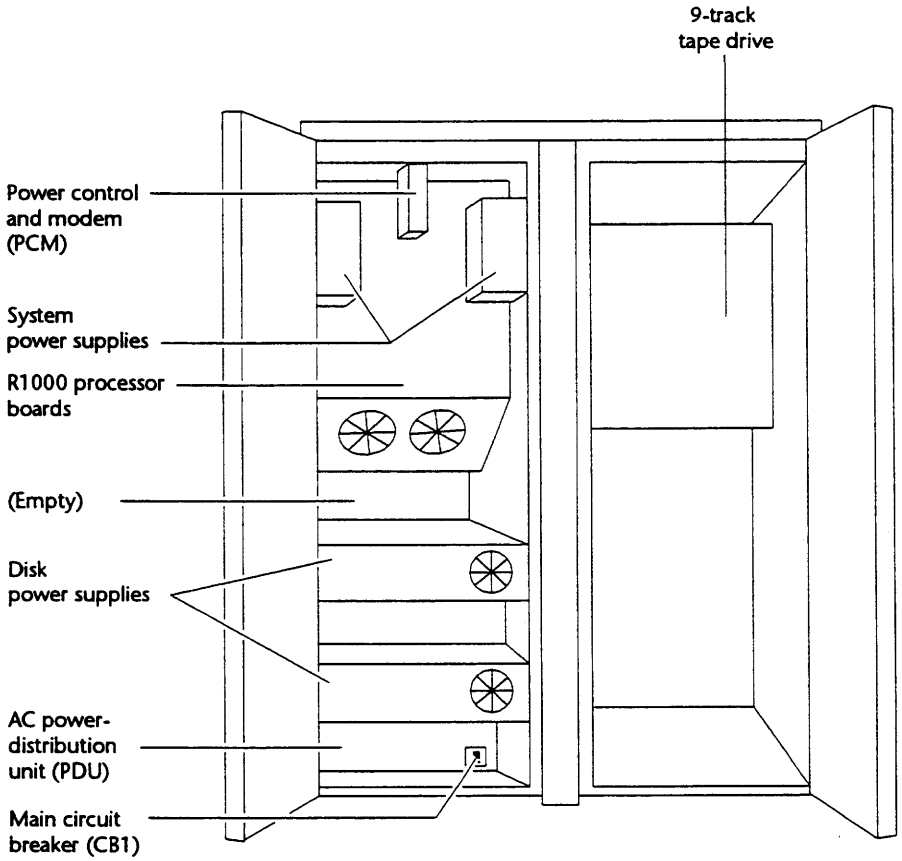

Figure 11-8 R1000 Series 200, Model 20B, Rear Interior View

Model 40

The Model 40 is two Model 20 systems mounted in one three- or four-bay stand-alone unit. The two R1000 processor units are mounted in separate CPU bays (CPU-0 and CPU-1), with one peripheral bay between the two CPU bays. In the four-bay Model 40, one optional expansion bay is added to the left of CPU-0. See Figure 11-9 for an illustration of the front of the Model 40 cabinet.

The peripheral bay between CPU-0 and CPU-1 can contain a 9-track tape drive, one or two 10-inch disk drives, and one 16-port communications panel.

In the four-bay Model 40, the expansion bay can contain a 9-track tape drive, one or two 10-inch disk drives, and one 16-port communications panel.

The Model 40 supports the Rational Environment.

The Model 40's RS232C communication ports are installed in two to four communications panels, which are mounted in the bottom of two to four bays. The printer, user terminals, and communication servers are connected in these communication ports. Each communications panel can support up to 16 user terminals or communication servers, although usually one port (port 31) in each CPU bay's communication panel is reserved for the printer.

See Figure 11-11 for a detailed illustration of the communications panel. The Model 40's control panels are located along the top of the CPU bays' front doors. See Figure 11-12 for a detailed illustration of the control panel.

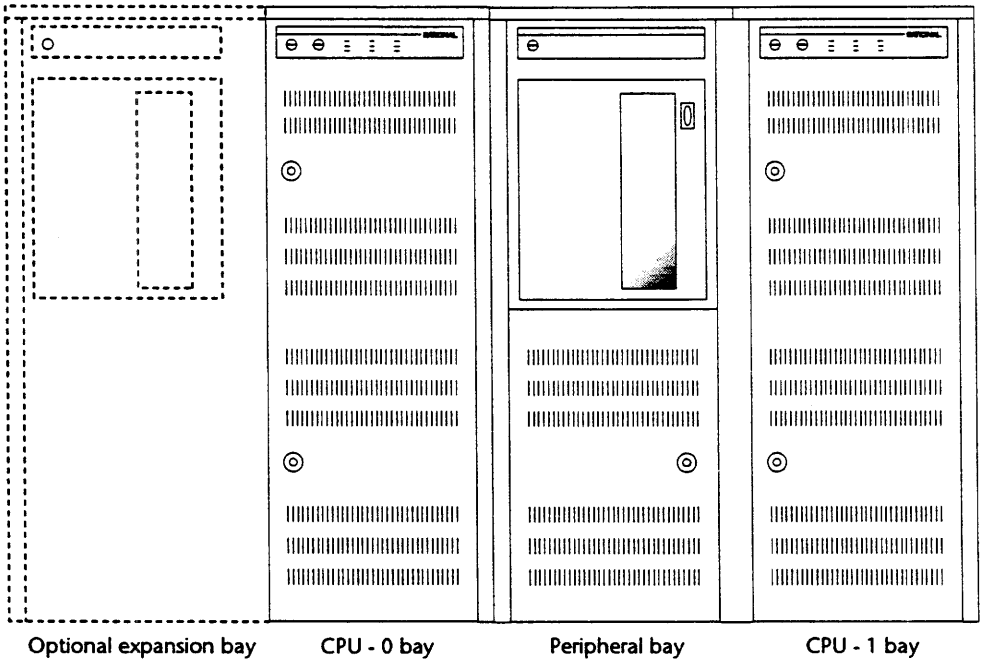

Figure 11-9 Series 200, Models 40 and 40B

Model 40B

The Model 40B is two Model 20B systems mounted in one three- or four-bay standalone unit. The two R1000 processor units are mounted in separate CPU bays (CPU-0 and CPU-1), with one peripheral bay between the two CPU bays. In the four-bay Model 40B, one optional expansion bay is added to the left of CPU-0. See Figure 11-9 for an illustration of the front of the Model 40B cabinet.

The peripheral bay between CPU-0 and CPU-1 can contain a 9-track tape drive and one 16-port communications panel.

In the four-bay Model 40B, the expansion bay can contain a 9-track tape drive and one 16-port communications panel.

The Model 40B supports the Rational Environment.

The Model 40B's RS232C communication ports are installed in two to four communications panels, which are mounted in the bottom of two to four bays. The printer, user terminals, and communication servers are connected in these communication ports. Each communications panel can support up to 16 user terminals or communication servers, although usually one port (port 31) in each CPU bay's communication panel is reserved for the printer.

See Figure 11-11 for a detailed illustration of the communications panel. The Model 40B's control panels are located along the top of the CPU bays' front doors. See Figure 11-12 for a detailed illustration of the control panel.

TAPE/PRINTER SWITCH UNIT ON MODELS 40 AND 40B

This section describes the R1000 Series 200 tape/printer switch unit that is installed on Model 40 machines.

Purpose of Tape/Printer Switch

When this switch unit is installed, the two CPUs in a Model 40 machine can share a single tape drive and a single Rational printer.

Description of Tape/Printer Switch

The tape/printer switch unit includes a tape-drive keyswitch mounted on the front of a peripheral bay and a tape/printer switch mounted in the rear of the same peripheral bay (see Figure 11-10).

The tape/printer switch unit contains two separate switches, one for routing tape-drive signals and the other for routing printer signals:

- Tape-drive switch section: A ribbon cable from each CPU is connected to the tape/printer unit. The position of the keyswitch on the front of the peripheral bay controls the tape-drive signal-routing switch in the switch unit.
- Printer switch section: A multiconductor cable from port 31 of each CPU's communication panel is connected to the tape/printer switch unit. Software signals from the CPUs control the printer signal-routing switch in the switch unit.

Equipment Configurations

All Model 40 configurations include one or two tape/printer switch units.

Three-Bay Model 40

In the three-bay Model 40, the switch unit is mounted in the top rear side of the peripheral bay (see Figure 11-10). This switch is accessible by opening the peripheral bay's rear door. When you are facing the front of the system, note that the CPU bay to the left of the tape drive is CPU-0 and the CPU bay to the right of the tape drive is CPU-1.

Four-Bay Model 40

A Model 40 can be configured with an optional expansion bay to the left of the CPU-0 bay (see Figure 11-10). The optional expansion bay is added to a three-bay Model 40 when the following configuration requirements exist:

- A second 9-track tape drive is installed on the system
- A fourth disk drive is installed for either CPU (see Chapter 16)
- A second communications panel is installed for CPU-0

If two 9-track tape drives are installed on the system, a second tape/printer switch also will be installed. This second switch assembly is mounted in the top rear side of the optional expansion bay.

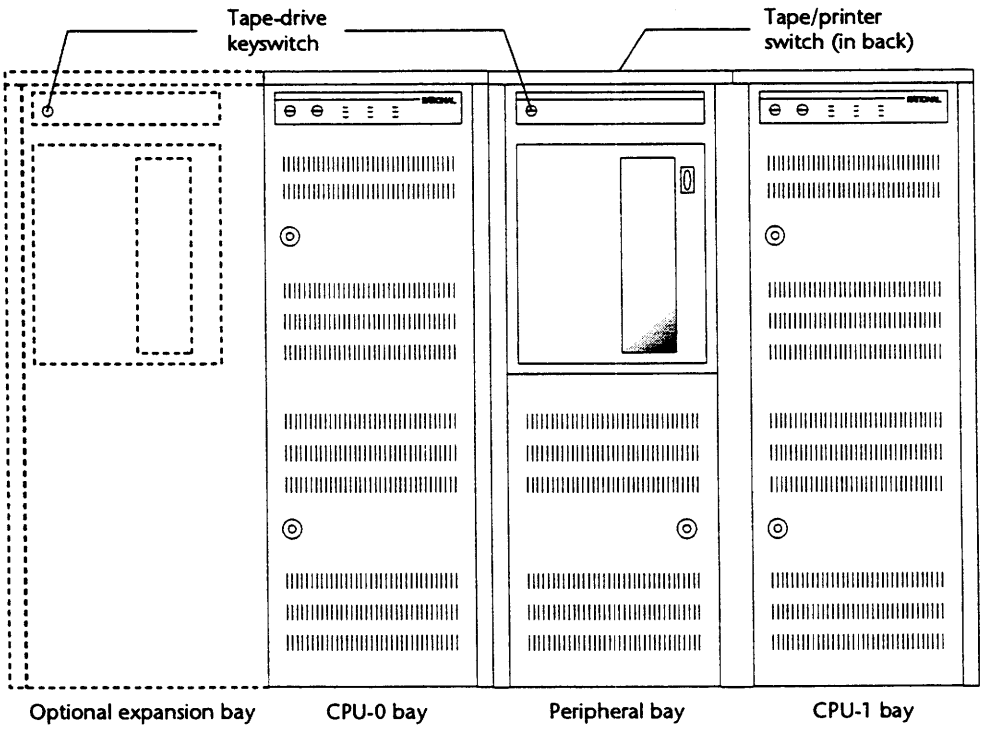

Figure 11-10 Location of Tape-Drive and Tape/Printer Switches on Model 40

Setting the Tape-Drive Keyswitch

All Model 40 configurations include one or two front-panel keyswitches that allow you to connect a selected CPU with a selected tape drive.

Caution: *When the tape-drive keyswitch has connected one of the CPUs, but you want the other CPU connected, you must rotate the keyswitch to the other position. Otherwise, the desired CPU "sees" the tape drive as being offline. However, DO NOT change the position of the tape-drive keyswitch while a CPU is using the tape drive. (For example, do not move the keyswitch after a mount request has been handled and the tape is being read/written.) Before changing the position of the tape-drive keyswitch, always check the tape drive. If a tape is loaded in the tape drive, you probably should assume that the tape drive is in use.*

Three-Bay Model 40

The tape-drive keyswitch is mounted on the left side of the peripheral bay's control panel, above its 9-track tape drive (see Figure 11-10). This keyswitch can be switched to the positions labeled CPU-0 or CPU-1. When the keyswitch is in the CPU-0 position, the 9-track tape drive is connected to CPU-0 and disconnected from CPU-1. When the keyswitch is in the CPU-1 position, the 9-track tape drive is connected to CPU-1 and disconnected from CPU-0. Note that the connected CPU "sees" the tape drive as online, but the disconnected CPU "sees" the tape drive as offline and therefore cannot access it.

Four-Bay Model 40

If there is a second 9-track tape drive installed in the system, it is installed in the optional expansion bay. The tape-drive keyswitch for this tape drive is mounted on the left side of the optional expansion bay's control panel, above its 9-track tape drive (see Figure 11-10). When there are two 9-track tape drives on the system, the normal mode for operation is for the tape-drive keyswitch on the optional expansion bay to be in the CPU-0 position and the tape-drive keyswitch on the peripheral bay to be in the CPU-1 position.

Printer Portion of the Switch Unit

The tape/printer switch unit also allows both CPU-0 and CPU-1 to share a single Rational printer. This is done by connecting the CPU-0 printer port (usually port 31 in the CPU-0 bay) to the CPU0-I/O port on the tape/printer switch unit, and the CPU-1 printer port (usually port 31 in the CPU-1 bay) to the CPU1-I/O port on the tape/printer switch unit. The printer is connected to the printer port on the tape/printer switch unit.

Output to the printer is hardware-flow-controlled by the switch unit. When one of the two CPUs has a job to be printed, the switch allows that CPU to begin printing the job. That CPU has control of the printer switch until the job is completed. Once the job is complete, the printer switch is free to allow another connection from the next CPU that has a job to be printed.

When a tape/printer switch unit is mounted in the optional expansion bay and only one Rational printer is shared by both CPUs, nothing is connected to the printer connector on this tape/printer switch unit.

COMMUNICATIONS PANEL

Figure 11-11 illustrates the communications panel, showing the panel switches, the status light-emitting diodes (LEDs), and the numbering arrangement of the ports on the panel, labeled 16 through 31 from left to right. Note that:

- Port 16 is reserved for the operator's Environment terminal.
- Port 31 is reserved for the Rational printer.
- All other ports are available for user terminals and communication servers.

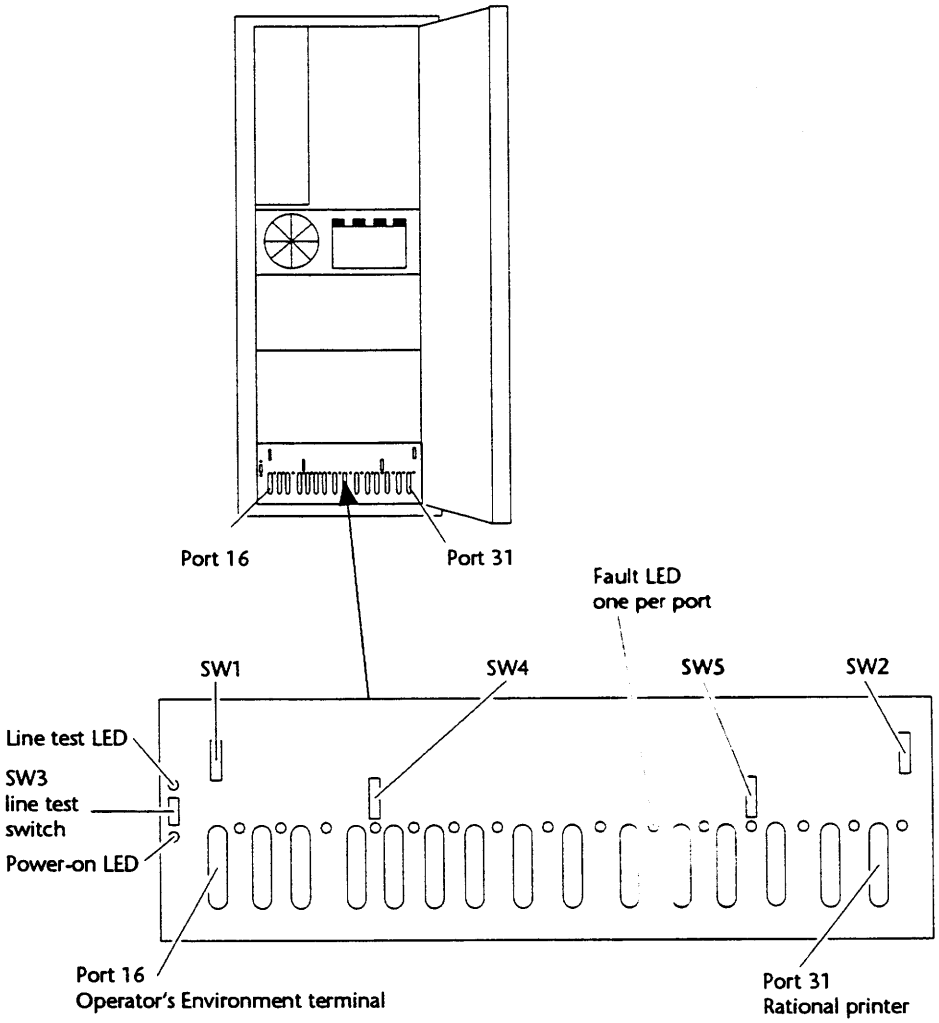

Figure 11-11 Series 200 Communications Panel

CPU BAY CONTROL PANEL

In general, the controls and indicators on this panel pertain to various aspects of starting and stopping the system, displaying the state and activity of components in the system, and performing low-level diagnostic and configuration functions. The CPU bay control panel is shown in Figure 11-12.

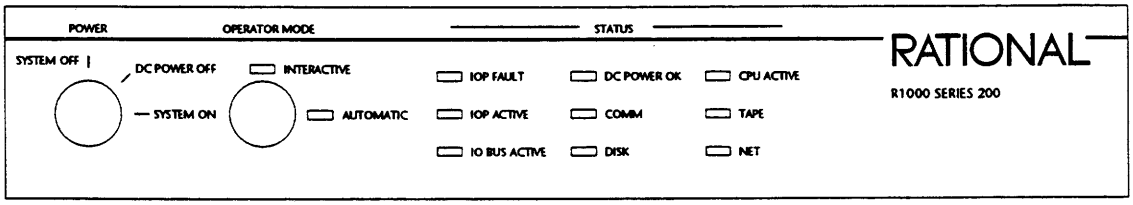

Figure 11-12 Series 200 Control Panel

- **Power**

  The power keyswitch allows you to switch AC and DC power on and off.

  - **System Off:** Turning the power keyswitch to the System Off position removes most of the R1000 electronics from AC power without tripping the main circuit breaker (CB1). Because the main cooling fans are "unswitched"—that is, connected to CB1 but not to the power keyswitch—they will continue to run. Note that turning the keyswitch to the System Off position before correctly shutting down the Environment can cause recent work to be lost. For the correct shutdown procedure, see Chapter 2, "Stopping and Starting the System."
  - **DC Power Off:** Turning the power keyswitch to the DC Power Off position removes DC power from the R1000 processor boards but leaves the AC power on for the disk drives and other AC-powered components (see Chapter 2 and Chapter 16). Leaving the AC power on keeps the fans blowing air over the components, reducing potential heat stress on system components.
  - **System On:** Turning the power keyswitch to the System On position applies normal AC and DC power to the system.

- **Operator Mode**

  The operator-mode keyswitch provides access to various configuration parameters that define the boot process. Your Rational technical representative normally sets these parameters during the installation process, tailoring your system's configuration to your specific site requirements.

  - **Automatic:** When the keyswitch is turned to Automatic, the boot process proceeds according to preset parameters. For most installations, these parameters result in the standard boot process described in the "Description of the Standard Boot Process" section in Chapter 2.
  - **Interactive:** When the keyswitch is turned to Interactive, the boot process pauses at the following menu and then proceeds according to the selection:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

- **Status**

  The nine status LEDs indicate several kinds of R1000 operation states, as described below:

  - The **IOP Fault** LED, when illuminated, indicates that an unresolved problem occurred during an I/O processor operation.
  - The **IOP Active** LED, when flickering, indicates normal I/O processor operations.
  - The **IO Bus Active** LED, when flickering or illuminated, indicates activity on the I/O bus.
  - The **DC Power OK** LED, when illuminated, indicates that all DC power supplies are powered on and are outputting correct DC voltages.
  - The **Comm** LED, when flickering or illuminated, indicates activity between a communications device and the processor.
  - The **Disk** LED, when flickering or illuminated, indicates activity between a disk drive and the processor.
  - The **CPU Active** LED, when flickering or illuminated, indicates that activity is occurring within the R1000 CPU.
  - The **Tape** LED, when flickering or illuminated, indicates activity between a tape drive and the processor.
  - The **Net** LED, when flickering or illuminated, indicates activity between the Ethernet network and the processor.

  For a description of the normal operation of these LEDs, especially during system startup and shutdown, see Chapter 2.

POWER CONTROL AND MODEM (PCM) UNIT

The PCM unit, mounted in the top rear of the CPU bay, contains eight indicator LEDs that can be useful for system diagnostics (see Figure 11-13). Even if a customer is at a restricted site that does not allow dial-out, the PCM unit can be used by the system manager to make a preliminary diagnosis of a system failure. The functions of each LED, connector, and switch follow:

- **+15 V OK:** When this green LED is illuminated, the power supply is within the satisfactory range (+15 V ±5%). Note that under normal operating conditions this is the only LED that is illuminated.
- **PS 0 Fault:** When this red LED is illuminated, a fault has been detected in power supply 0.
- **PS 1 Fault:** When this red LED is illuminated, a fault has been detected in power supply 1.
- **–15 V Fault:** When this red LED is illuminated, the power supply is not within the satisfactory range (–15 V ±5%).
- **EPO:** When this red LED is illuminated, the I/O processor has initiated emergency power-off.
- **Over-Temp:** When this red LED is illuminated, the system has overheated because of a failure in the cooling system.
- **Power Margin** When this amber LED is illuminated, the voltage margin has been set 5% higher or lower (for diagnostics only).

- **Clock Margin:** When this amber LED is illuminated, the clock margin has been set 5% higher or lower (for diagnostics only).
- **Modem:** This connector is an RJ45 phone jack for diagnostics.
- **White button:** This push-button switch, located on the bottom edge of the PCM board, resets the R1000.

  Caution: *Press this button only when advised to do so by Rational personnel.*

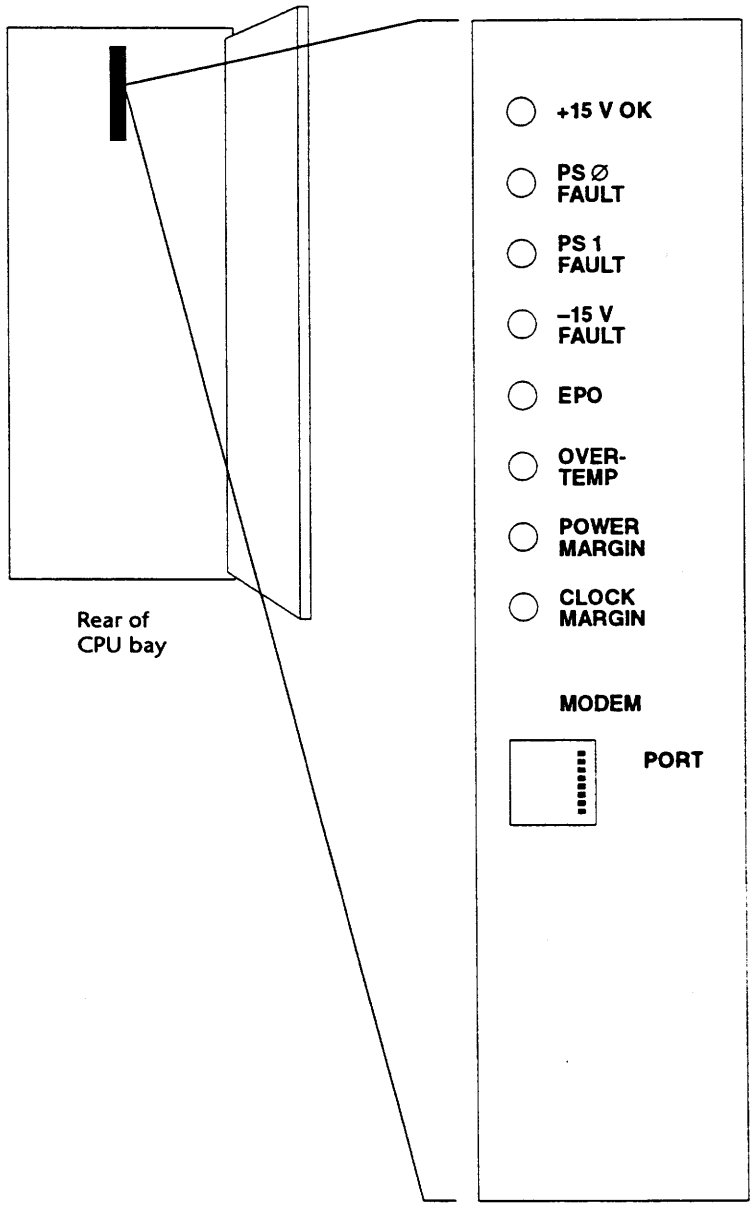

Figure 11-13 Series 200 Power Control and Modem (PCM) Unit

The modem part of the PCM is used for two purposes: dial-out and diagnostics. In dial-out mode, the R1000 uses this modem to notify Rational of a system crash or certain disk problems. When the R1000 has crashed and is in diagnostics mode, the modem gives Rational support personnel remote access to the R1000

operator's console. This modem also allows a remote debugger to be connected for debugging problems in the Environment. Note that the modem does *not* give access to an Environment session.

POWER-DISTRIBUTION UNIT (PDU)

The PDU is mounted on the bottom of the CPU bay. The PDU is accessible when the rear door of a CPU bay is open. (For the location of the PDU, refer to Figure 11-14.)

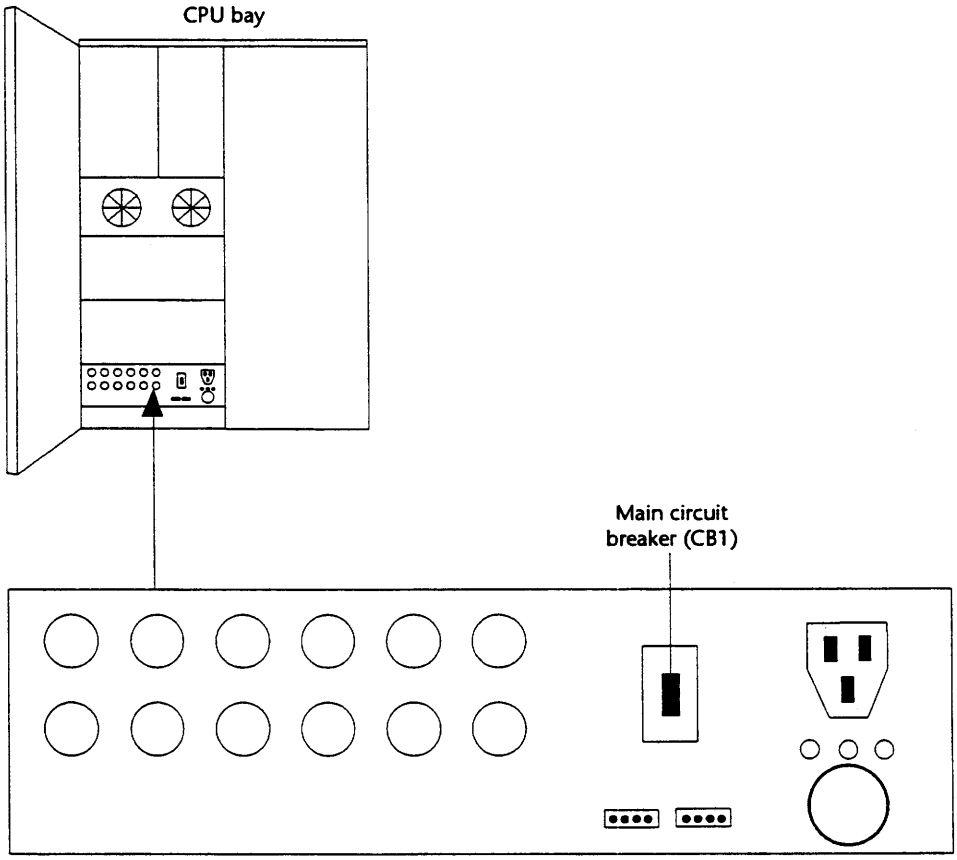

Figure 11-14 Series 200 Power-Distribution Unit (PDU)

Each PDU includes the following features:

- Main circuit breaker (CB1)
- DC power-control switch
- Remote switch connector
- Service power/monitor receptacle
- Switched outlets for each power supply and disk drive
- Unswitched outlets for the fans and each tape drive
- Other switched and unswitched outlets
- Three fuses (F1, F2, and F3)

12 Overview of Series 300 System

This chapter provides an overview of the Rational R1000 Series 300 System (300S) and Series 300 Coprocessor (300C). Both of these systems are network resources. The 300S is a two-cabinet, standalone unit—one cabinet contains the CPU and the other contains up to four disk drives. The Series 300C is a single-cabinet, diskless CPU that works in conjunction with widely used file servers. The 300C is connected to the file server by a coprocessor link.

SERIES 300S CONFIGURATION

As shown in Figure 12-1, a Series 300S consists of a CPU bay (part number 120-002917) and a peripheral bay (part number 120-002980).

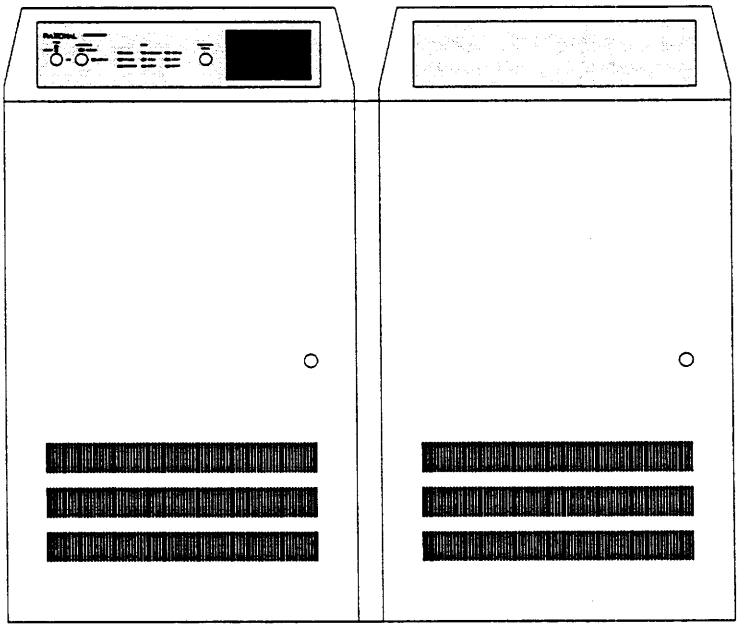

Figure 12-1 Series 300S

The 300S CPU bay (see Figures 12-2 and 12-3) contains:

- R1000 processor and memory boards (four 8 Mb, a single 32 Mb, or two 32 Mb)
- Controllers for I/O devices
- DC power supplies
- System control panel

- 8-mm tape drive
- Two RS232 ports, labeled "Operator's Console" and "External Modem"
- One Ethernet port
- AC power-distribution unit

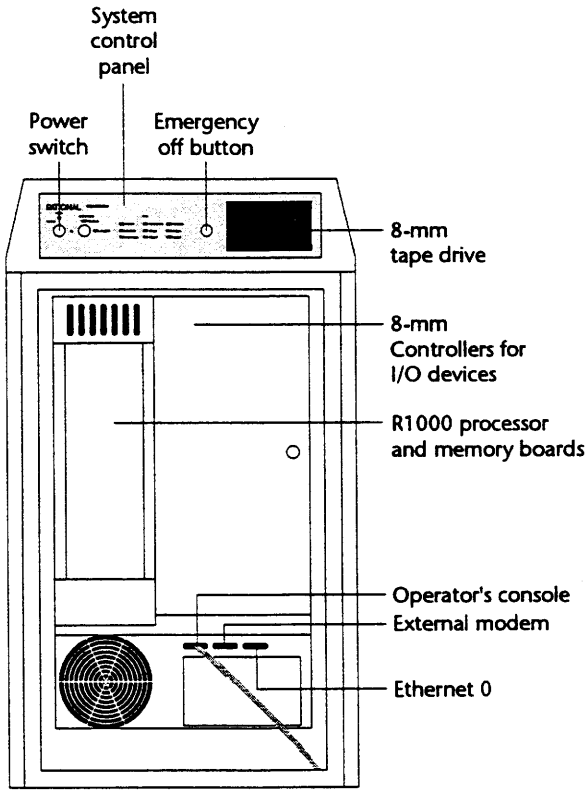

Figure 12-2 Series 300S CPU Bay, Front View

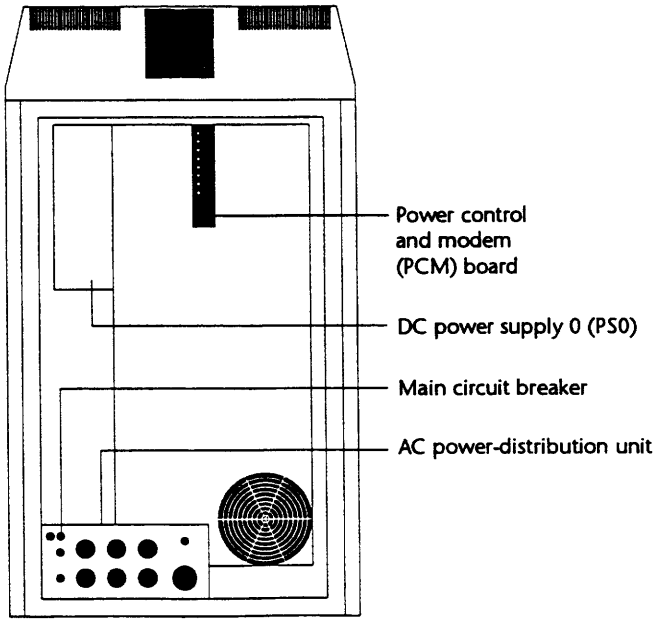

Figure 12-3 Series 300S CPU Bay, Rear View

The 300S peripheral bay (see Figures 12-4 and 12-5) contains:

- Disk housing and disk units
- DC power supply
- 9-track tape drive (optional, not shown)

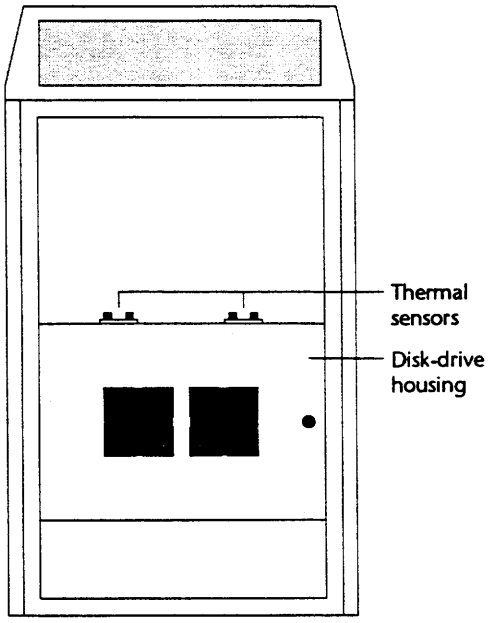

Figure 12-4 Series 300S Peripheral Bay, Front View

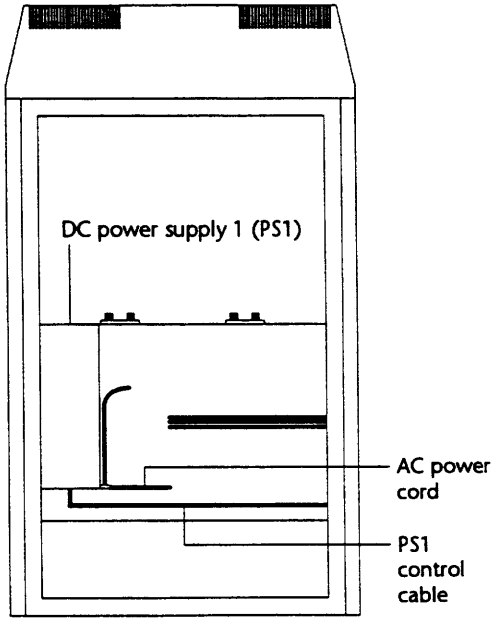

Figure 12-5 Series 300S Peripheral Bay, Rear View

SERIES 300C CONFIGURATION

The Series 300C (Figure 12-6) is a diskless configuration that works with file servers. In this configuration, the 300C is the *client*, and the computer that provides disk-storage space is the *server*. The client and the server are connected by a dedicated Ethernet, called the *coprocessor link*. The client runs the Rational Environment software. The server runs its native operating system (for example, the IBM AIX™ system), plus coprocessor server software provided by Rational. See Figure 12-7.

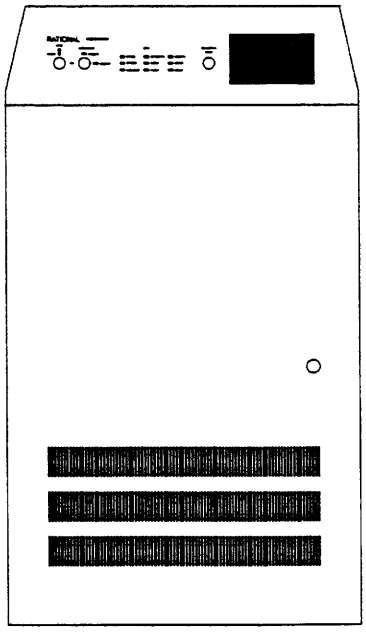

Figure 12-6 Series 300C

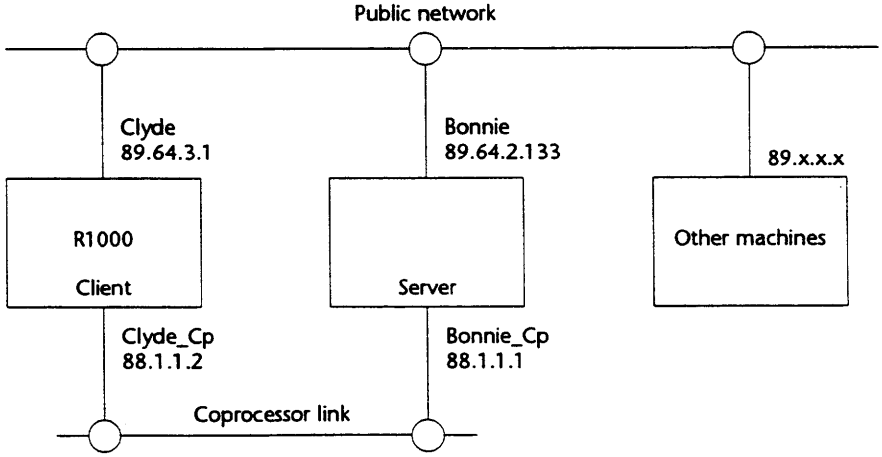

Figure 12-7 Networking for Client and Server

A Series 300C CPU bay (see Figures 12-8 and 12-9) contains:

- R1000 processor and memory boards
- Controllers for I/O devices
- DC power supplies
- System control panel
- 8-mm tape drive
- Two RS232 ports, labeled "Operator's Console" and "External Modem"
- Two Ethernet ports
- AC power-distribution unit

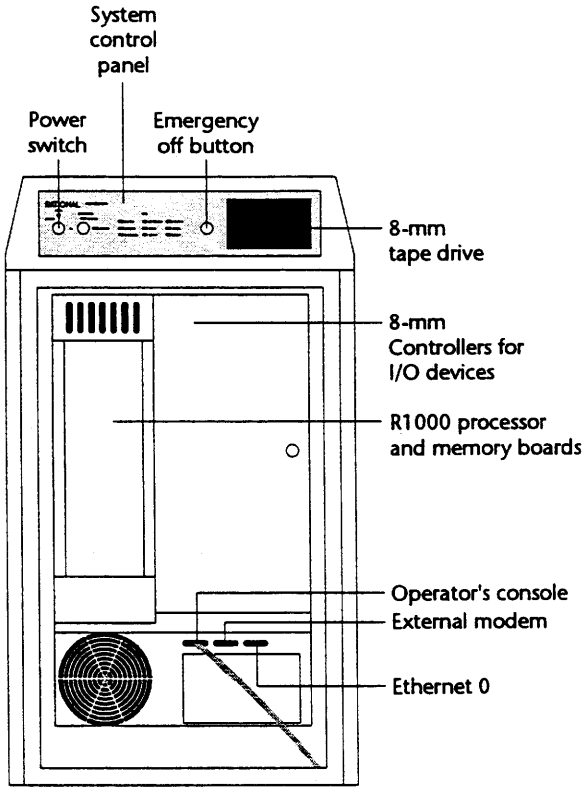

Figure 12-8 Series 300C CPU Bay, Front View

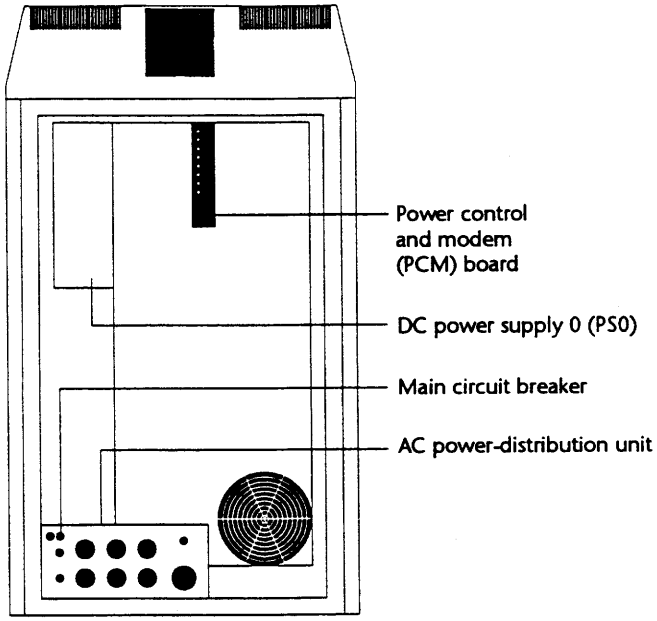

Figure 12-9 Series 300C CPU Bay, Rear View

POWER-DISTRIBUTION UNIT (PDU)

When you open the rear door of the Series 300S or 300C CPU bay (see Figures 12-3 and 12-9), notice the PDU located at the bottom of the bay (see Figure 12-10). This unit contains:

- AC/DC power-control switch
- PS0/PS1 inhibit
- Main circuit breaker (CB1)
- Fuse (F1)
- PS0 receptacle
- PS1 receptacle
- Tape-drive receptacle
- Two spare receptacles
- Fans receptacle
- Auxiliary circuit breaker (CB2)
- Service/power monitor

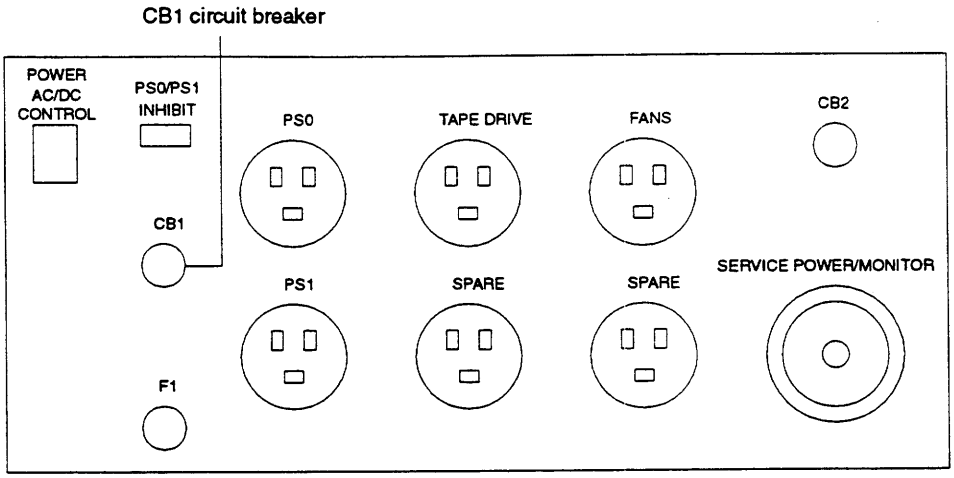

Figure 12-10 Series 300 Power-Distribution Unit (PDU)

SYSTEM CONTROL PANEL

The system control panel is the same for the 300S and 300C. In general, the controls and indicators on this panel pertain to various aspects of starting and stopping the system, determining the basic state of the system and I/O activity, and performing low-level diagnostic and configuration functions. The system control panel is shown in Figure 12-11.

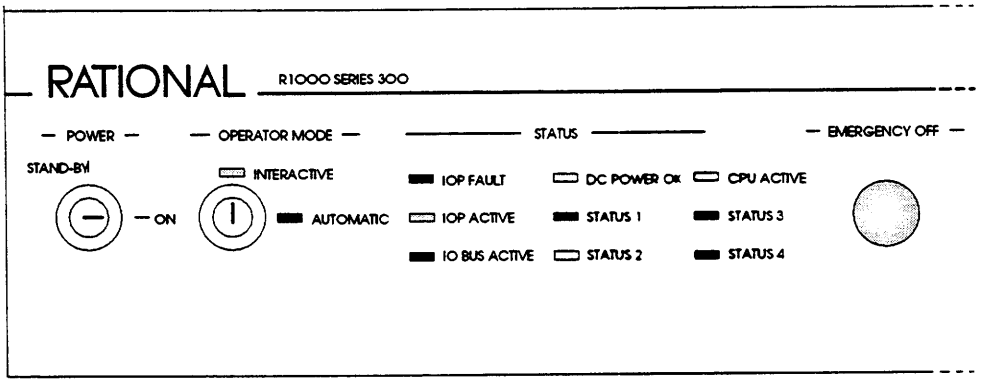

Figure 12-11 Series 300 System Control Panel

- **Power**

  Switching the power keyswitch to the Standby position removes AC power to the R1000 without tripping the main circuit breaker. Turning the power off without properly shutting down the Environment can cause recent work to be lost (see Chapter 2, "Stopping and Starting the System").

  When the power keyswitch is switched to the middle (unlabeled) position, AC power is applied to the power supplies and all CPU fans begin blowing. This reduces potential heat stress on system components by continuing to circulate air through the system. Note that when the power keyswitch is in the middle position, DC power is still removed from the system and the logic circuitry has been shut off. For the Series 300S, this means that the disk-drive fans do not operate when the power keyswitch is in the middle position.

  Switching the power keyswitch to the On position applies DC power to the system. The disk-drive fans begin operating on a Series 300S.

- **Operator Mode**

  The operator-mode keyswitch provides access to various configuration parameters that define the boot process. Your Rational support representative normally sets these during the installation, tailoring the configuration to your specific site requirements.

  - **Automatic:** When the keyswitch is turned to Automatic, the boot process proceeds according to preset parameters. For most installations, these parameters result in the standard boot process described in the "Description of the Standard Boot Process" section in Chapter 2.
  - **Interactive:** When the keyswitch is turned to Interactive, the boot process pauses at the following menu and then proceeds according to the selection:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

- **Status**

  The nine status light-emitting diodes (LEDs) indicate several kinds of R1000 operation states, as described below:

  - The **IOP Fault** and **IOP Active** LEDs indicate the state of the I/O processor.
  - The **IO Bus Active** LED indicates activity on the I/O bus.
  - The **DC Power OK** LED indicates that all DC power supplies are powered on and outputting DC voltage.

- The **CPU Active** LED indicates activity occurring within the R1000 CPU.
- The Comm (or **Status 1**), Disk (or **Status 2**), Tape (or **Status 3**), and Net (or **Status 4**) LEDs indicate that activity is occurring between the processor and communications devices, disk drive, tape drive, or Ethernet, respectively.

Under normal operating conditions after power-up, the IOP Active, IO Bus Active, Disk (Status 2), and CPU Active LEDs flicker frequently, sometimes remaining completely on or off for short periods of time. The DC Power OK LED should always be on, and the IOP Fault LED should always be off.

POWER CONTROL AND MODEM (PCM) BOARD

The PCM board, located in the top rear of the CPU bay, contains eight LEDs that can be useful for system diagnostics (see Figure 12-12). Even if a customer is at a restricted site that does not allow dial-out, the PCM board can still be used by the system manager to make a preliminary diagnosis of a system failure. The panel lights correspond to the following functions:

- **+15 V OK:** When this green LED is illuminated, the power supply is within the satisfactory range (+15 V ±5%). Note that under normal operating conditions this is the only LED that is illuminated.
- **PS 0 Fault:** When this red LED is illuminated, a fault has been detected in power supply 0.
- **PS 1 Fault:** When this red LED is illuminated, a fault has been detected in power supply 1.
- **–15 V Fault:** When this red LED is illuminated, the power supply is not within the satisfactory range (–15 V ±5%).
- **EPO:** When this red LED is illuminated, the I/O processor has initiated emergency power-off.
- **Over-Temp:** When this red LED is illuminated, the system has overheated because of a failure in the cooling system.
- **Power Margin** When this amber LED is illuminated, the voltage margin has been set 5% higher or lower (for diagnostics only).
- **Clock Margin:** When this amber LED is illuminated, the clock margin has been set 5% higher or lower (for diagnostics only).
- **Modem:** This connector is an RJ45 phone jack for diagnostics.
- **White button:** This push-button switch, located on the bottom edge of the PCM board, resets the R1000.

  Caution: *Press this button only when advised to do so by Rational personnel.*

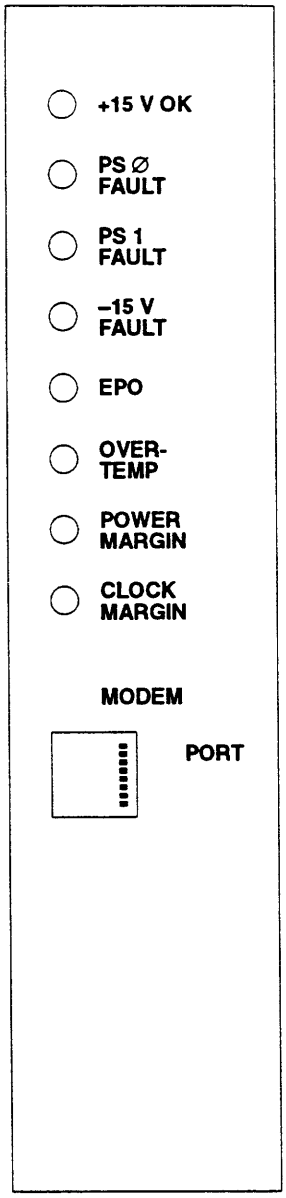

Figure 12-12 Series 300 PCM Board

DISK-DRIVE CONTROL PANEL (300S ONLY)

The disk unit's control panel is shown in Figure 12-13. To gain access to the disk unit, you will need to open the door of the disk enclosure. The control panel contains a write-protect switch and indicators with which to operate the drive, described below. Note that each disk drive has a disk-drive control panel.

- **Power On (indicator)**

  This indicator is lit when the DC power-supply unit is functioning and the unit is turned on.

  The disk drives do not power-up until the main R1000 power is turned on.

  When power is applied to the drive, rotation of the spindle motor is enabled. Spinup takes approximately 50 seconds, at which time the Ready lamp lights. Spindown takes approximately 15 seconds after the power switch is set to the Off position.

- **Ready (indicator)**

  The Ready indicator lights when the spindle has reached the rated speed and no fault condition exists in the drive. It also lights when the initial seek is performed or when a seek or RTZ (return-to-zero) operation has completed.

- **Protect (indicator)**

  This indicator lights when write operations have been inhibited.

- **Check (indicator)**

  The Check indicator lights when any fault condition has occurred. If this condition occurs, contact the Rational Response Center (see Appendix D).

- **PTCT (write-protect switch)**

  When set to the On position, the PTCT switch inhibits write operations to the disk. This switch normally should be in the Off position.

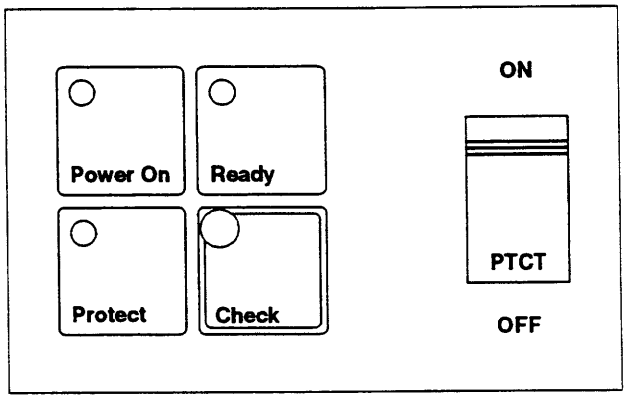

Figure 12-13 Series 300S Disk-Drive Control Panel

13 Overview of Series 400 System

This chapter provides an overview of the Rational R1000 Series 400. The Series 400 is a single-cabinet unit that contains a central processing unit (CPU) with 32 megabytes of memory (with the option of upgrading to 64 megabytes). The Series 400 can be configured to contain one to four disks (the Series 400 System or 400S) or as a diskless coprocessor (the Series 400 Coprocessor or 400C). The coprocessor works in conjunction with widely used file servers and is connected to the file server by a coprocessor link (refer to the "Series 400C Configuration" section for more information about the coprocessor configuration).

Customers decide which configurations best suit the needs of their sites; all configurations are designed to be used on a network.

SERIES 400S CONFIGURATION

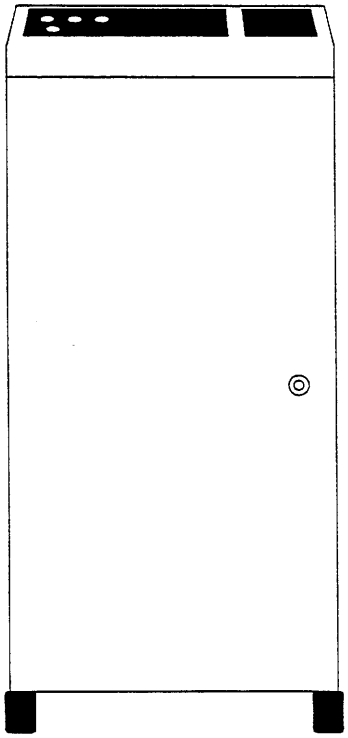

Figure 13-1 Series 400 Cabinet, Front View

The Series 400S is a processor that supports the Rational Environment. As shown in Figure 13-1, the base configuration of the Series 400 is a standalone unit that contains one to four disks. The front and back cabinet doors can be locked to

protect internal hardware from damage and to protect users from hazards such as electric shock.

As shown in Figures 13-2 and 13-3, the Series 400 contains:

- System control panel
- 8-mm tape drive
- R1000 boards (foreplane access to processor and memory boards)
- An input/output panel containing:
  - Two RS232 ports: Comm and operator's console
  - One or two network ports: Ethernet 0 and Ethernet 1
  - Other assorted device interfaces, switches, and status LEDs (not shown in Figures 13-2 and 13-3—see Figure 13-7 for more detail)
- Disk housing
- Fan housing
- DC power supply
- AC power-distribution unit
- Backplane (access to R1000 boards)

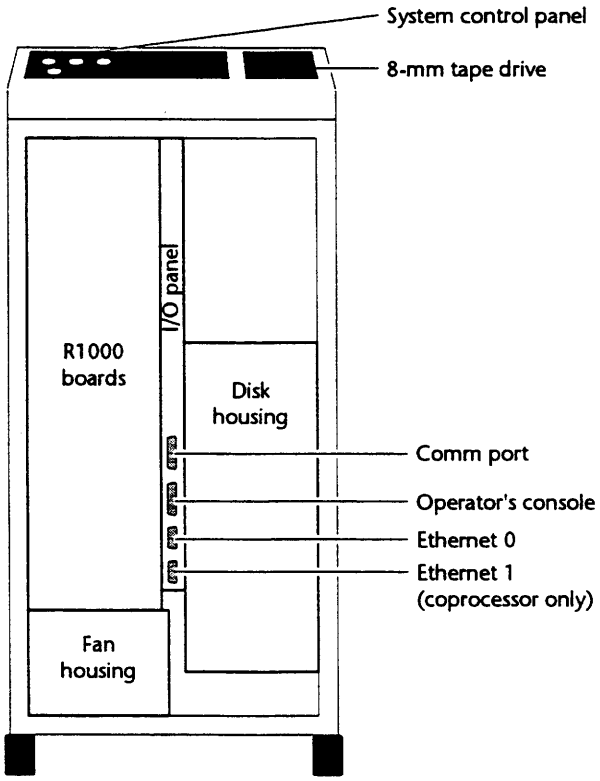

Figure 13-2 Series 400 Interior, Front View

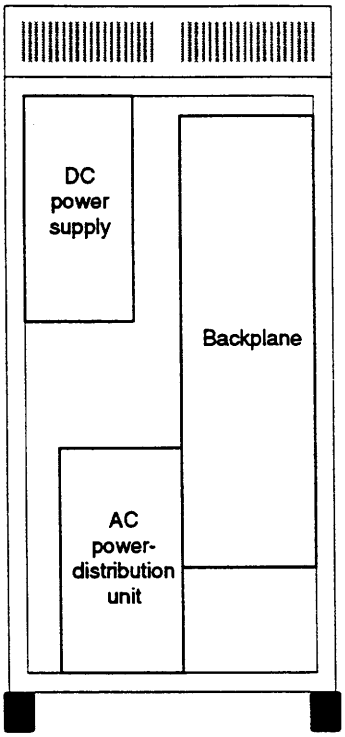

Figure 13-3 Series 400 Interior, Back View

SERIES 400C CONFIGURATION

The Series 400C is a diskless configuration that works with file servers. In this configuration, the coprocessor is the *client*, and the computer that provides disk-storage space is the *server*. The client and the server are connected by a dedicated Ethernet, called the *coprocessor link*. Figure 13-4 shows one type of coprocessor-link configuration.

The client runs the Rational Environment software. The server runs its native operating system (for example, IBM AIX), plus coprocessor server software provided by Rational. Rational provides either Series 400 IBM RISC System/6000 server software (product number 3000-00638) or Series 400 Sun Workstation server software (product number 3000-00639).

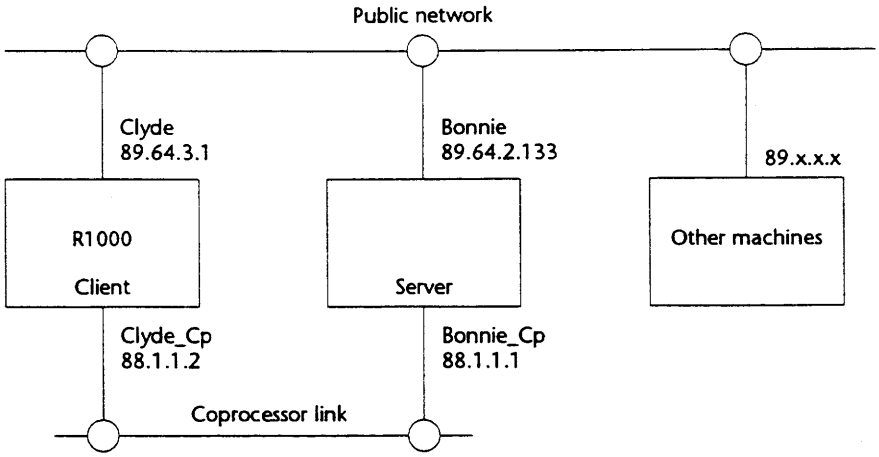

Figure 13-4 Networking for Client and Server

POWER-DISTRIBUTION UNIT (PDU)

When you open the rear door of the R1000 Series 400 cabinet, notice the PDU located at the bottom of the cabinet (Figure 13-5). This unit contains:

- 250 V, 25 A fuse (F1)
- Auxiliary circuit breaker (CB2)
- Main circuit breaker (CB1)
- Power supply receptacle (PS0)
- Spare (service power/monitor)
- Service power/monitor
- Power AC/DC control switch

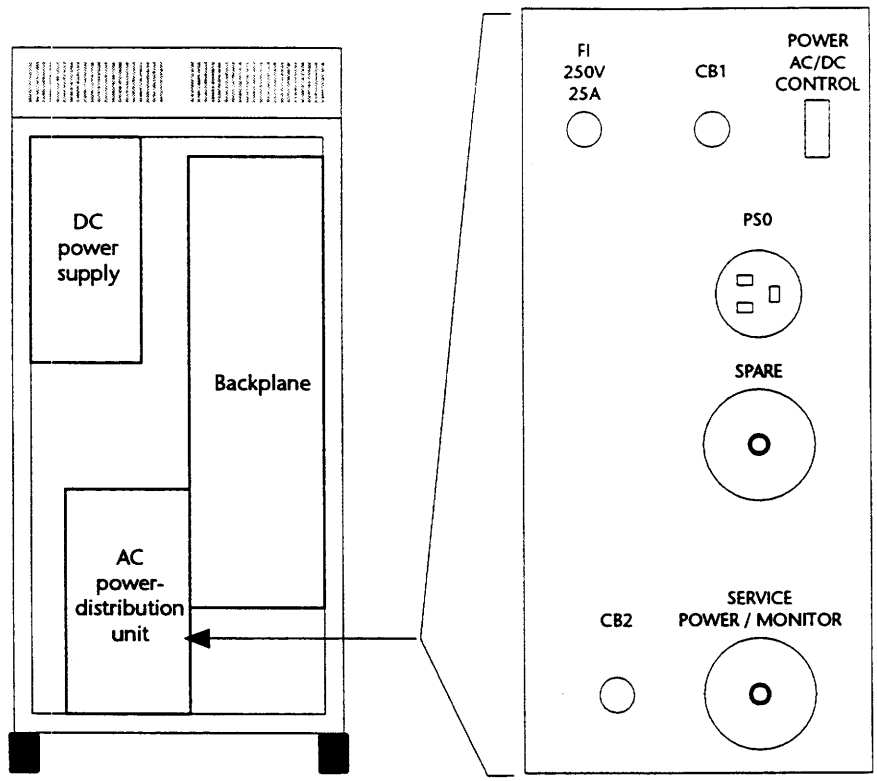

Figure 13-5 Series 400 Power-Distribution Unit (PDU)

SYSTEM CONTROL PANEL

In general, the controls and indicators on the system control panel are used for starting and stopping the system, determining the basic state of the system and I/O activity, and performing low-level diagnostic and configuration functions. The system control panel is shown in Figure 13-6. Also shown in this figure is the 8-mm tape drive (refer to Chapter 15 for a detailed description of the 8-mm tape drive).

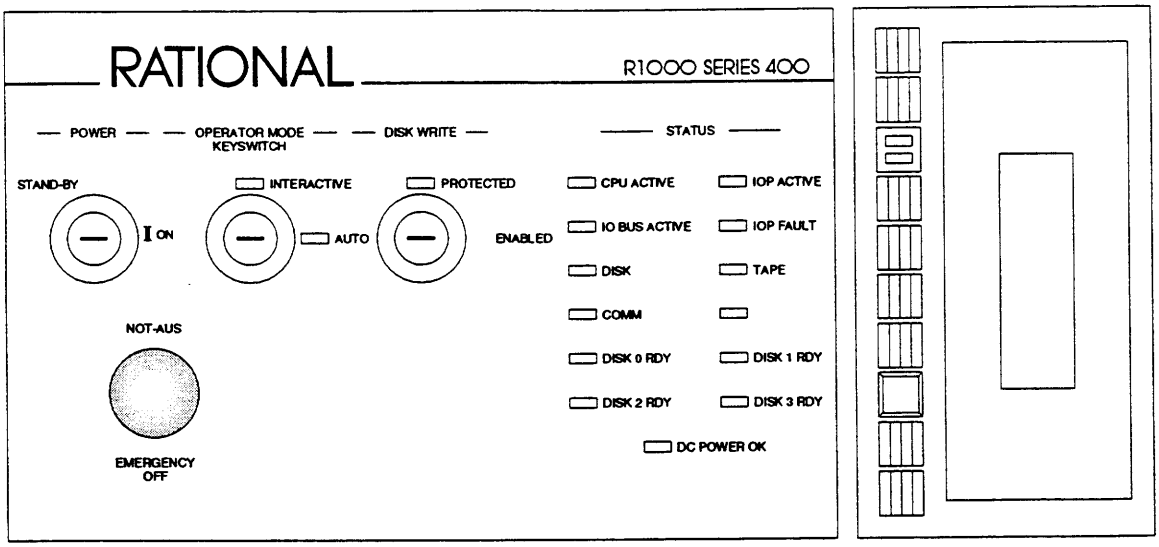

Figure 13-6 Series 400 Control Panel and 8-mm Tape-Drive Panel

The features of the system control panel are described below.

- **Power**

  Switching the power keyswitch to the Standby position removes AC power to the R1000 without tripping the main circuit breaker. Doing so without properly shutting down the Environment can cause recent work to be lost (see Chapter 2, "Stopping and Starting the System").

  Switching the power keyswitch to the On position applies AC power to the PDU and DC power to the DC power supply. The AC-powered CPU fans begin to blow (on a Series 400S, the DC-powered disk-drive fans also begin to blow).

- **Operator Mode**

  The operator-mode keyswitch provides access to various configuration parameters that define the boot process. Your Rational technical representative normally sets these parameters during the installation process, tailoring your system's configuration to your specific site requirements.

  - **Automatic:** When the keyswitch is turned to Automatic, the boot process proceeds according to preset parameters. For most installations, these parameters result in the standard boot process described in the "Description of the Standard Boot Process" section in Chapter 2.
  - **Interactive:** When the keyswitch is turned to Interactive, the boot process pauses at the following menu and then proceeds according to the selection:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system

Enter option [Boot System] :
```

- **Disk Write**

  The disk-write switch is used to write-protect disks located in the disk housing. When switched to the Protected position, the disk-write switch prevents write operations to the disk. This switch normally should be in the Enabled position.

  Note: *If the Series 400 has been configured as a coprocessor, this switch does not write-protect server disks.*

- **Status**

  The thirteen status light-emitting diodes (LEDs) indicate several kinds of R1000 operation states, as described below:

  - The **IOP Fault** and **IOP Active** LEDs indicate the state of the I/O processor.
  - The **IO Bus Active** LED indicates activity on the I/O bus.
  - The **DC Power OK** LED indicates all DC power supplies are powered on and are providing acceptable DC voltage.
  - The **CPU Active** LED indicates activity within the R1000 CPU.
  - The **Comm**, **Disk**, **Tape**, and **Net** LEDs indicate activity between the processor and communications devices, disk drive, tape drive, or Ethernet, respectively.

  Under normal operating conditions after power-up, the IOP Active, IO Bus Active, Disk, and CPU Active LEDs flicker frequently, sometimes remaining completely on or off for short periods of time. The DC Power OK LED should always be on, and the IOP Fault LED should always be off.

- The **Disk 0 Rdy**, **Disk 1 Rdy**, **Disk 2 Rdy**, and **Disk 3 Rdy** indicators light up when disk units 0–3, respectively, have reached the rated speed and no fault condition exists in the drives. Each LED also lights up when the initial seek is performed or when a seek or RTZ (return-to-zero) operation has completed. Note that these indicator LEDs do not indicate when server disks are ready if the Series 400 is configured as a coprocessor.

  The disk drives do not power-up until the main R1000 power is turned on and the system begins to boot. During reboot after power-on, each disk drive is spun up, starting with the highest-numbered disk unit (for example, disk unit 2 if a system has three disk units). After the first disk unit reaches maximum speed, the next disk is spun up, and so forth. Each disk takes about 50 seconds to spin up, at which time the Rdy indicators light. Spindown takes approximately 15 seconds after the power switch is set to the Standby position.

I/O PANEL

The I/O panel, which runs vertically along the middle front of the interior of the CPU cabinet, contains two Ethernet ports (Ethernet 0 and Ethernet 1), an operator's console port, a communications (comm) port (used to access the Environment using the RS232C protocol), an EPROM write-protect switch and status LED, a reset button, a built-in modem that allows dial-out capability, and seven status LEDs that can be useful for system diagnostics (see Figures 13-2 and 13-7). Even if a customer is at a restricted site that does not allow dial-out, the I/O panel can still

be used by the system manager to make a preliminary diagnosis of system problems. The EPROM write-protect switch, status LEDs, and ports are described in greater detail below.

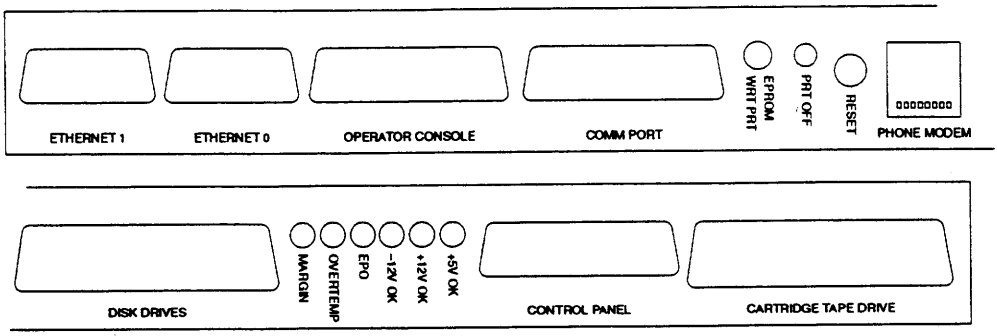

Figure 13-7 Series 400 I/O Panel

- **+5 V OK:** When this green LED is illuminated, it indicates that the DC power supply is operating properly. Under normal operating conditions, this LED should always be on.
- **+12 V OK:** When this green LED is illuminated, it indicates that the DC power supply is operating properly. Under normal operating conditions, this LED should always be on.
- **–12 V OK:** When this green LED is illuminated, it indicates that the DC power supply is operating properly. Under normal operating conditions, this LED should always be on.
- **EPO:** When this red LED is illuminated, it indicates that the IO processor has initiated emergency power-off.
- **Overtemp:** When this red LED is illuminated, it indicates that the system has overheated.
- **Margin:** When this amber LED is illuminated, it indicates that the DC voltage or clock speed has been set 5% higher or lower (for diagnostics only).
- **Phone Modem:** This is an RJ45 telephone jack, used for diagnostics.
- **Reset (white button):** Located next to the built-in telephone modem, this button is used to reset the R1000.

  Caution: *Press this button only when advised to do so by Rational personnel.*
- **Prt Off:** When this yellow LED is illuminated, it indicates that the EPROM write-protect switch is off. Under normal circumstances, this status LED should not be illuminated.
- **EPROM Wrt Prt:** This switch protects the EPROMs from electrical surges and should be on under normal circumstances (if this switch is off, the yellow Prt Off LED is illuminated).

The ports labeled "disk drives," "control panel," and "cartridge tape drive" are used to connect the I/O panel to the disk drive, system control panel, and the 8-mm cartridge drive, respectively.

14 Printer Operations and Maintenance

This chapter describes:

- Components and controls in the Rational printer
- Normal operation of the printer
- Selecting the printer's operating mode
- Diagnosing and recovering from printer errors
- Routine maintenance of the printer, such as loading paper and changing ribbons
- Configuring the system's printer port
- Setting up the print spooler

Illustrations in this chapter will help you locate and identify components and controls on the Rational printer.

PRINTER COMPONENTS

This section describes some of the printer components necessary in day-to-day operations. To locate the external components, refer to Figure 14-1. To locate the internal components, refer to Figure 14-2.

External Components

The main-line power switch on the right side of the printer switches on and off all power to the printer.

The forms rack in the back collects the printed paper.

The front control panel allows the operator to monitor current printer status and to change printer-setup parameters.

The rear panel control on some models is for the operator's convenience.

The top cover, when opened, allows the operator to access internal components.

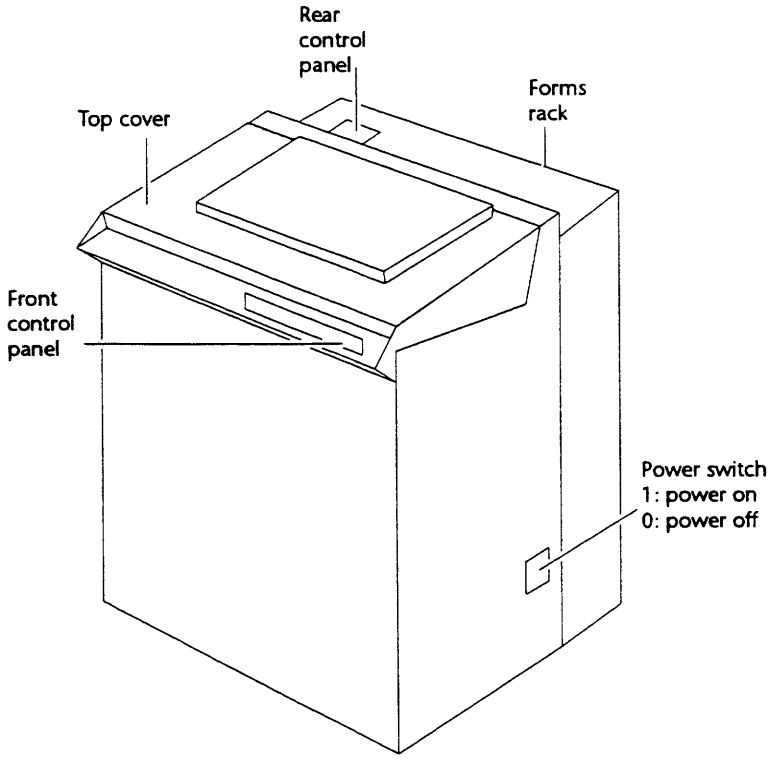

Figure 14-1 Printer Exterior Components

Internal Components

The tractor release levers secure the horizontal alignment of the forms. They should be released to reset the horizontal position of the forms or to adjust lateral tension in the forms.

The advance-forms knob aligns the forms with the line scale. It can be used to realign the vertical print position.

The throat-open lever unlocks the print unit and increases the clearance between the print and base units. Open the throat-open lever when aligning forms or changing ribbons. When this lever is open, the control panel continually displays `Intlk` until the lever is locked.

The band-position knob (under the band cover) adjusts the vertical position of the pulley. Use this knob to align the print-band position.

The band-exchange knob (under the band cover) releases the print band. It is used only when changing the print band.

The forms-thickness knob adjusts both the print timing and the print pressure to compensate for the thickness of the forms. The forms-thickness setting may need adjusting if you notice:

- Missed characters
- Fading at the tops or bottoms of characters

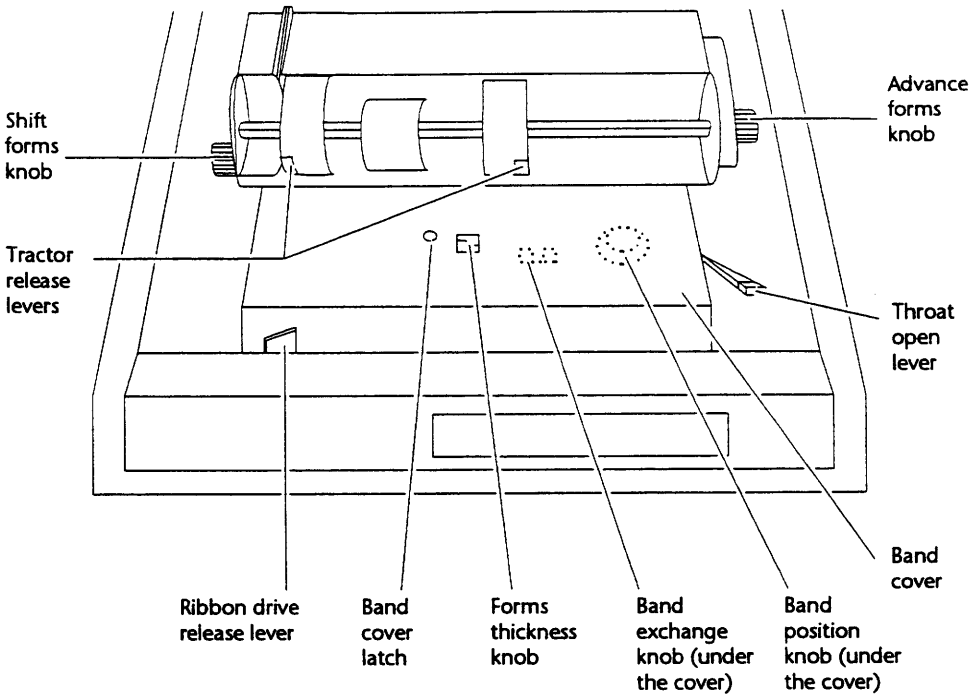

Figure 14-2 Printer Interior Components

- Skewed forms
- Torn perforations

Turn the forms-thickness knob counterclockwise to compensate for thick forms and clockwise to compensate for thin forms.

The ribbon-drive release lever releases the ribbon for changing. When turned counterclockwise, it disengages the ribbon-feed rollers. It should be reset before the printer is returned online. When the ribbon-drive release lever is open, the Status indicator on the control panel displays `Ribbon`, and the Error Reset button must be pressed to return the printer to normal operation.

The band-cover latch secures the band cover and should be released when changing or cleaning the print band.

The shift-forms knob changes the lateral position of the forms by ±1 millimeter. It should be used for fine horizontal adjustment of the forms.

PRINTER FRONT-PANEL CONTROLS

The Rational Printer front-panel controls include switches, light-emitting diodes (LEDs), and a status display, as shown in Figure 14-3. These features are described below, in order from right to left as they appear on the control panel. The features toward the right of the panel are described in Table 14-1, "Basic Printer Controls." The remaining features are described in Table 14-2, "Status, Error, and Mode Controls."

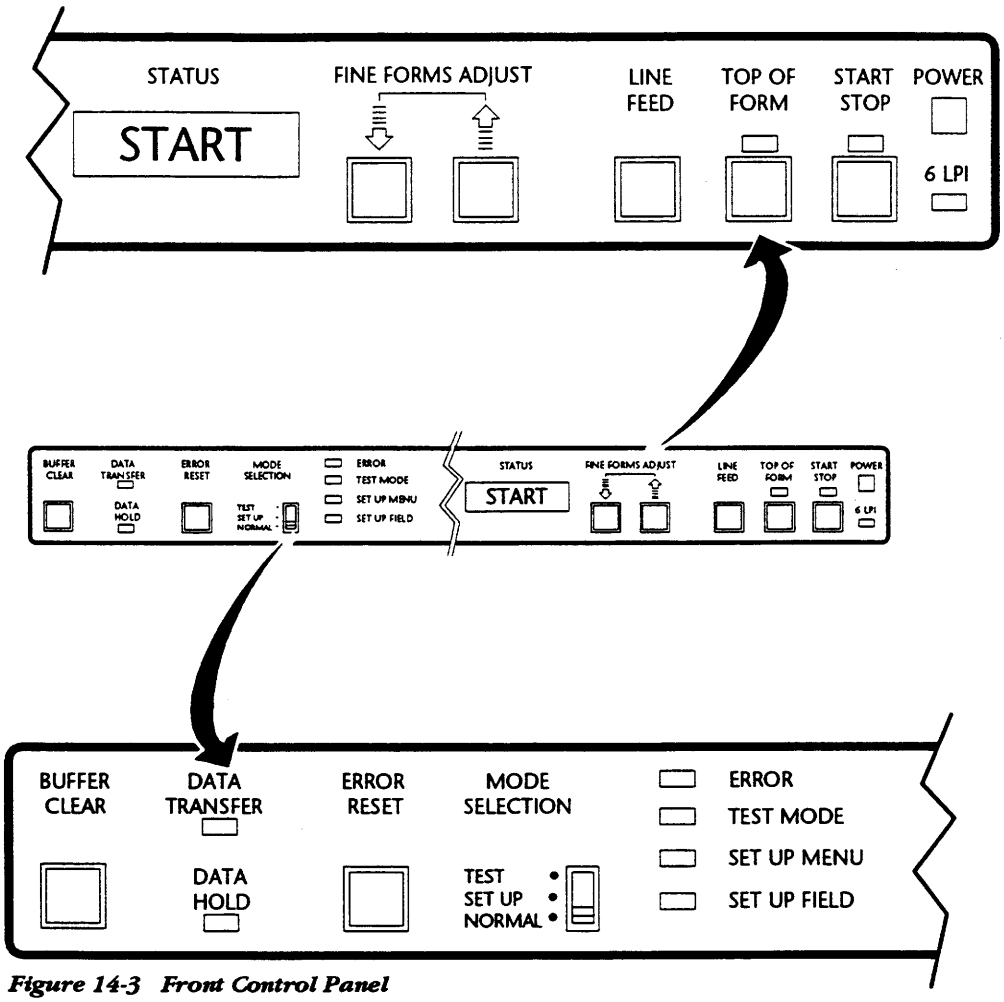

Figure 14-3 Front Control Panel

Basic Printer Controls

Table 14-1 describes the functions of the basic printer controls located on the right side of the printer's front control panel.

Table 14-1 Basic Printer Controls

LEDs and Controls	Functions
Power	The Power LED is on when the power is on and off when the power is off. See Figure 14-1 for the location of the main-line power switch that turns the power on and off.
6 LPI	The 6 LPI LED is on when the form-feed pitch is six lines per inch and off when the form-feed pitch is eight lines per inch.
Start/Stop	The Start/Stop button starts and stops printer operation. When started, the printer is online and can receive data; when stopped, the printer is offline and cannot receive data. Successive presses of the button toggle between starting and stopping. Status indicator displays START if you have started the printer and STOP if you have stopped it. The Start/Stop LED is located just above the Start/Stop button. This LED is on whenever you start the printer and off whenever you stop it. In test mode, the Start/Stop button starts and stops test execution. In setup mode, the button is used to choose a particular setup menu or field.
Top of Form	The Top of Form button feeds the current form to the top position. If the printer is online, pressing the Top of Form button takes the printer offline, performs four form feeds, and then automatically returns it to online after 20 seconds. The Top of Form LED is on when print forms are positioned at the top.
Line Feed	The Line Feed button feeds the print form one line at a time. If the button is held down, it continues to feed the print form until released. This button is disabled when the printer is online.
Fine Forms Adjust	The Fine Forms Adjust buttons make fine vertical adjustments to the form position when the printer is offline. The buttons are disabled when the printer is online in normal mode. When the [↑] button is held down, the form is fed slowly upward. When the [↓] button is held down, the form is fed slowly downward. When adjusting the form downward, be sure that the paper is loaded without slack. Open the throat-open lever, and pull the form until tight. When the printer is in test mode or setup mode, the Fine Forms Adjust buttons toggle between menus and fields.

Status, Error, and Mode Controls

Table 14-2 describes the functions of the status, error, and mode controls located on the left side of the printer's front control panel.

Table 14-2 Status, Error, and Mode Controls

LEDs and Switches	Functions
Status	The Status indicator displays the printer's current state. These states are summarized in Table 14-3. Certain states indicate various normal operations. Other states result from errors that need to be corrected. When one of the latter states is displayed, the Error LED is on.
Error	The Error LED is on when an error has been detected. Check the Status indicator to see where the error occurred.
Test Mode	The Test Mode LED is on when the printer is in test mode and is off when the printer is in either normal or setup mode.
Set Up Menu	The Set Up Menu LED is on when the printer is in setup mode and the Status indicator contains the name of a setup menu item. The LED is off at all other times.
Set Up Field	The Set Up Field LED is on when the printer is in setup mode and the Status indicator contains the name of a setup field item. The LED is off at all other times.
Mode Selection	The Mode Selection switch toggles between test, setup, and normal operating modes. When you set the switch to another mode, the Status indicator displays the name of that mode. The three modes are discussed in "Operating Modes," below.
Error Reset	The Error Reset button resets the printer so that it can resume normal operations after an error has been corrected. The Status indicator returns to the state that was displayed before the error occurred.
Data Transfer	The Data Transfer LED is on while data is actually being transferred between the R1000 and the printer.
Data Hold	The Data Hold LED is on while the printer buffer contains data.
Buffer Clear	The Buffer Clear button empties transferred data from the printer buffer.

PRINTER REAR-PANEL CONTROLS

Some models of the Rational printer also have an abbreviated control panel in the rear that is easy to reach when removing printouts. (For the location of this panel, see Figure 14-1.) The rear control panel contains a second Top of Form button and a second Start/Stop button. The function of these buttons is identical to the function of the corresponding buttons on the front control panel.

OPERATING MODES

The Rational printer has the following three operating modes:

- *Normal mode:* The printer operates online and prints data from the system.
- *Test mode:* The printer operates offline for executing test printing and self-diagnostics.
- *Setup mode:* The printer allows you to select and change basic operating features such as line-feed pitch or page length.

Normal Mode

To operate the printer in normal mode:

1. Confirm that the form is in place and the top cover is closed.
2. Set the Mode Selection switch to Normal.
3. Turn on the power by setting the power switch to the on position: 1. The Power LED lights and the Status indicator displays DIAG while the printer automatically runs its diagnostic program.
   - If no errors are detected, the Status indicator displays STOP.
   - If an error is detected, the Error LED lights and the Status indicator displays an error status. For more information, see "Error Recovery in Normal Mode" and Table 14-3, below.
4. When the Status indicator displays STOP, press the Start/Stop switch. The indicator now displays START.

During normal operation, pressing Start/Stop takes the printer offline or returns it to online. Also, if the printer is online, pressing the Top of Form button takes the printer offline, performs four form feeds, and then automatically returns it to online after 20 seconds.

Error Recovery in Normal Mode

When an error occurs:

- The Start/Stop LED turns off and the Error LED lights.
- The Status indicator displays an error status.
- The Start/Stop button is disabled.

To recover from an error:

1. Check Table 14-3 for the meaning of the error status displayed in the Status indicator.
2. Correct the problem yourself or contact your Rational support representative. (See Appendix D.)

3. Press the Error Reset button after correcting the problem. If there are no more errors, the Error LED turns off and the Status indicator displays STOP.
4. Press the Start/Stop switch. The Status indicator displays START.

Table 14-3 Printer Status Codes

Status Display	Explanation
BAND	The printer detected a print-band fault. Use the Fine Forms Adjust buttons to display error contents. Then contact your Rational support representative. (See Appendix D.)
COVER	The top cover is open. Close and secure the cover. Press the Error Reset button when ready to continue normal operations.
DIAG	The printer is executing the self-diagnosis program.
FAN	The printer fan is malfunctioning. Contact your Rational support representative. (See Appendix D.)
FCTERR	The printer detected errors in the FCT unit. Contact your Rational support representative. (See Appendix D.)
FORM	The printer is out of paper. Load paper. Check the paper path. Press the Error Reset button when ready to continue normal operations.
FUSE	The printer detected a blown fuse. Contact your Rational support representative. (See Appendix D.)
HMRCHK	Hammer check. Contact your Rational support representative. (See Appendix D.)
INTLK	The band-cover latch or the throat-open lever was left open or inadequately locked. Close and secure the latch or lever. Press the Error Reset button when ready to continue normal operations.
JAM	The printer detected an abnormal formfeed. Paper may be jammed or torn. Check the paper path, and adjust the paper if necessary. Press the Error Reset button when ready to continue normal operations.
LDERR	Errors were detected while loading FCB data. Use the Fine Forms Adjust buttons to display error contents. Then contact your Rational support representative. (See Appendix D.)
NOCHNL	The printer cannot feed forms because it cannot find the channel specified by the skip command. Contact your Rational support representative. (See Appendix D.)
RIBBON	The printer detected abnormal ribbon movement, or the ribbon-drive release lever may be open. Check both the lever and the ribbon. Reset or replace a jammed or torn ribbon. Make sure the lever is closed and secured. Press the Error Reset button when ready to continue normal operations.
SET UP	The printer is operating in setup mode.
STACKR	The stacker is full and needs to be emptied.
START	The printer is online.
STOP	The printer is offline.
SYNCH	Contact your Rational support representative. (See Appendix D.)
TEST	The printer is operating in test mode.

Test Mode

Enter test mode when you want to check print quality or verify normal printer operations. Table 14-4 describes the tests you can run in test mode.

Table 14-4 Printer Test Menu

Option	Code	Name	Function
1	SFTPRT	Shift printing	Instructs the printer to print a rotating character pattern that tests all characters. Shifts all letters on the print band. Executes all checks. If an error is detected, the printer stops.
2	FIXPRT	Fix printing	Prints the character specified with FIXDT in all print positions.
3	FIXDT	Print data	Selects the character to be printed with FIXPRT.
4	NONPRT	Nonprint	Checks the print and feed cycles without printing.
5	PSEADJ	Adjust phase	Fine-tunes the print line during FIXPRT. Use the Fine Form Adjust buttons to adjust the print line.
6	BNDTST	Band test	Tests the operation of the print band and ribbon.
7	RBNTST	Ribbon test	Tests the operation of the ink ribbon.
8	FDTST	Vertical feed	Checks form-feed control by executing sequential form feeds without printing. Checks for jamming.
9	SET FL	Form length	Changes the test program to form-length control for resetting form length.
10	L.LOOP		Do not run this test. If you do, the Error LED will light.

Entering Test Mode

To enter test mode:

- If the printer is online and functioning normally:
  1. Press the Start/Stop button to stop the printer.
  2. Set the Mode Selection switch to Test.
  3. Confirm that the Test Mode LED lights and that the Status indicator displays TEST.
- If the printer is online and the Error LED is on:
  1. Set the Mode Selection switch to Test.
  2. Confirm that the Test Mode LED lights and that the Status indicator displays TEST.
- If the printer is offline:
  1. Set the Mode Selection switch to Test.
  2. Confirm that the Test Mode LED lights and that the Status indicator displays TEST.

Running Tests in Test Mode

To run one or more tests from the printer's test menu:

1. Refer to Table 14-4 for the name(s) of the test(s) you want to run.
2. Display the name of the desired test in the Status indicator.
   a. Press the Fine Forms Adjust [↑] button to display the first test name.
   b. If the displayed test is not the one you want, press the Fine Forms Adjust button [↑] to display the next test or press [↓] to display the previous test. Tests are displayed in the order given in Table 14-4. Keep pressing either the [↑] button or the [↓] button until the test you want appears in the Status indicator. Repeatedly pressing either of these buttons cycles through the test menu.
3. To start the displayed test, press the Start/Stop button. If the printer encounters errors while running a test, a status code indicating the error is displayed in the Status indicator. See Table 14-3 for a list of status codes.
4. To stop a test, press the Start/Stop button a second time.
5. To repeat the same test, press the Start/Stop button a third time.
6. To run a new test, display and start it as described in steps 1 and 2 above.

Returning to Normal Mode from Test Mode

To return to normal mode when you are finished with testing:

1. Set the Mode Selection switch to the Normal position. You can set the switch at any time during test-mode operation. Tests in progress will be interrupted.
2. Confirm that the Test Mode LED turns off and that the Status indicator displays STOP.
3. Press the Start/Stop button to put the printer online.

Setup Mode

The Rational printer contains a menu of parameters that control various aspects of printer operation, such as changing the page length or the number of lines per inch. You can change some of these parameters to customize the printer for your installation. Other parameters are set by a Rational support representative so that the printer can be connected to the R1000.

Enter setup mode when you need to change or verify certain characteristics of printer operation.

Setup Menus and Fields

Each printer parameter is called a *setup menu item.* Some sample setup menu items are LPI (lines per inch), FORM-L (page length expressed in number of lines), or FORM-I (page length expressed in inches).

Every menu item is associated with a field of values or options. Each menu item is always set to one of the values within its field. The values in a field are called *setup field items.* For example, the setup field items for the menu item LPI include 6LPI and 8LPI.

Table 14-5 at the end of this section lists the menu items for the Rational printer and gives the corresponding field items. Note that certain menu items listed in Table 14-5 may not appear on your model of printer.

Optional and Required Settings

You can change many of the setup menu items to suit your needs. However, certain menu-item settings should not be changed, because they control how the Rational printer communicates with the R1000. The menu items with required settings are indicated in Table 14-5. Note that certain menu items listed in Table 14-5 may not appear on your model of printer.

There are two ways to enter setup mode, depending on whether you need to change optional settings or required settings. To change only optional settings, enter setup mode using the procedure described in "Entering Setup Mode to Change Optional Settings," below. When you enter setup mode this way, the required menu items are protected. You will not be able to reset them, because you will not be able to display any other field item than the current setting.

If a Rational support representative requests that you change one of the required settings, enter setup mode using the procedure described in "Entering Setup Mode to Change Required Settings," below.

Entering Setup Mode to Change Optional Settings

To enter setup mode to change optional settings:

- If the printer is online in normal mode:
  1. Press the Start/Stop button to stop the printer.
  2. Confirm that the Status indicator displays `STOP`.
  3. Set the Mode Selection switch to Set Up.
  4. Confirm that the Status indicator displays `SET UP`.
- If the printer is in test mode or offline in normal mode:
  1. Confirm that the Status indicator displays `STOP`.
  2. Set the Mode Selection switch to Set Up.
  3. Confirm that the Status indicator displays `SET UP`.
- If the printer is in test mode:
  1. Confirm that the Status indicator displays `TEST`.
  2. Set the Mode Selection switch to Set Up.
  3. Confirm that the Test Mode lamp is off and that the Status indicator displays `SET UP`.
- Continue as described in "Displaying and Changing Printer Settings," below.

Entering Setup Mode to Change Required Settings

To enter setup mode to change required settings (recommended only if requested by Rational support personnel):

1. Turn the printer power off by setting the main-line power switch to the off (0) position.
2. Set the Mode Selection switch to Set Up.
3. Turn the printer power on by setting the main-line power switch to the on (1) position.
4. Confirm that the Power LED is on and that the Status indicator displays `SET UP`.
5. Continue as described in the next subsection, "Displaying and Changing Printer Settings."

Displaying and Changing Printer Settings

1. Once the Status indicator displays `SET UP`, display the first setup menu item:
   a. Press the Start/Stop button while pressing the Error Reset button.
   b. Confirm that the Set Up Menu LED lights.
   c. Confirm that the Status indicator displays `LPI`. Note that LPI is the first item of the setup menu.
2. Find and display the desired menu item:
   a. If LPI is the menu item you want, proceed to step 3.
   b. If you want another menu item, press the Fine Forms Adjust [↑] button to display the next menu item or press [↓] to display the previous menu item. Menu items are displayed in the order given in Table 14-5. Keep pressing either the [↑] button or the [↓] button until the menu item you want appears in the Status indicator. Repeatedly pressing either of these buttons will cycle the display through the entire setup menu.
3. Once the Status indicator displays the desired menu item, display the field item that is currently set:
   a. Press the Start/Stop button.
   b. Confirm that the Set Up Field LED lights.
   c. Confirm that the Status indicator displays the current field item for the chosen menu.
4. To change the current setting, find and display the desired field item: press [↑] to display the next field item or press [↓] to display the previous field item.

   Keep pressing either [↑] or [↓] until the field item you want appears in the Status indicator. Repeatedly pressing either button will cycle through the entire field for that menu item.

   If pressing the Fine Forms Adjust buttons does not change the field item displayed in the Status indicator, then that setting is required and unchangeable. See "Optional and Required Settings," above.
5. Set the desired field item when it appears in the Status indicator:
   a. Press the Start/Stop button.
   b. Confirm that the Set Up Field LED turns off and that the Set Up Menu LED lights again.
   c. Confirm that the Status indicator displays the chosen menu item again.
6. To change another setting, repeat steps 2 through 5.

Returning to Normal Mode from Setup Mode

To return to normal mode:

1. Set the Mode Selection switch to the Normal position. You can set the switch at any time during setup-mode operation.
2. Confirm that the Set Up Menu LED goes off and that the Status indicator displays STOP.
3. Press the Start/Stop button to put the printer online.

Example: Changing the Number of Columns on a Page

If wide paper is used in the printer, you probably want the printer to print 132 columns per page instead of 80 columns. Because the number of columns on a page is controlled by an optional printer setting, use the following procedure to change the setting (which assumes that the printer is online in normal mode):

1. Press the Start/Stop button to stop the printer. Confirm that the Status indicator displays STOP.
2. Set the Mode Selection switch to Set Up. Confirm that the Status indicator displays SET UP.
3. Press the Start/Stop button while pressing the Error Reset button to display the first setup menu item. Confirm that the Set Up Menu LED lights and that the Status indicator displays LPI.
4. Press the Fine Forms Adjust [↑] button until the Status indicator displays COLUMN.
5. Press the Start/Stop button to display the current setup field item for COLUMN. Confirm that the Set Up Field LED lights and that the Status indicator displays 80COL (the current setting).
6. Press [↑] until the Status indicator displays 132COL.
7. Press the Start/Stop button to set this field item. Confirm that the Set Up Menu LED lights again and that the Status indicator displays COLUMN.
8. Set the Mode Selection switch to Normal. Confirm that the Set Up Menu LED turns off and that the Status indicator displays STOP.
9. Press the Start/Stop button to put the printer online. Confirm that the Status indicator displays START.

Note that certain menu items listed in Table 14-5 may not appear on your model of printer.

Table 14-5 Printer Setup Menus and Fields

Menu Name	Field Names	Initial Setting	Required/ Optional	Initial Explanation
LPI	6LPI	6LPI	Optional	Specifies six lines per inch.
	8LPI			Specifies eight lines per inch.
				If a Select Vertical Pitch command is transmitted by the host, the line-feed pitch is changed before setting.
FLSSEL	LINES	LINES	Optional	LINES allows you to specify page length by number of lines.
	INCHES			INCHES allows you to specify page length by number of inches.
FORM-L	LIN018–LIN199	LIN066	Optional	Specifies the number of lines per page. Enabled only if FLSSEL is set to LINES.
FORM-I*	ICH003–ICH022	ICH011	Optional	Specifies the number of vertical inches per page. Enabled only if FLSSEL is set to INCHES.
FORM-F*	0ICH	0ICH	Optional	Specifies measurement for form-length fine adjustment.
	1/6ICH			
	1/4ICH			
	1/3ICH			
	1/2ICH			
	2/3ICH			
	3/4ICH			
	5/6ICH			
SKIPOV	LIN000–LIN015	LIN006	Optional	Specifies the number of lines to be skipped from the perforated line.
BOF CH	NO	NO	Optional	NO specifies that bottom-of-form (BOF) channel is not used.
	CH 2–CH 12			Specifies the channel number of the BOF channel (from CH 2 through CH 12).
COLUMN	136COL	80COL	Optional	Specifies the number of columns to be printed per line.
	132COL			
	80COL			
TRNCT	DISABL	DISABL	Required	DISABL executes the space command as instructed.
	ENABLE			ENABLE truncates excess line feeds, allowing line feeds up to, but not past, the next top of form, if too many are received.
LOW → UP	DISABL	DISABL	Required	DISABL prints lowercase characters as lowercase characters.
	ENABLE			ENABLE translates lowercase to uppercase characters for printing. Allows usage of 48- or 64-character-set print band for 96-character-set code.
I1403	DISABL	DISABL	Required	DISABL performs line-feed function if the printer receives "skip to channel N" command while the print line is presently at the specified channel.
	ENABLE			ENABLE neither prints the current line nor feeds forms under the above condition.

*Menu items marked with an asterisk may not appear on your model of printer.

Table 14-5 Printer Setup Menus and Fields (continued)

Menu Name	Field Names	Initial Setting	Required/ Optional	Initial Explanation
DPBELT*		BELT1		
D-MENU*		DISP		
L-END	NLonRM	NLonRM	Optional	NLonRM causes printer to execute new line on right margin.
	NO			NO does not cause printer to execute new line on right margin.
CRCODE	CR<>NL	CR<>NL	Optional	Carriage return does not equal new line.
	CR=NL			Carriage return is equal to new line.
L.TERM	PRTCMD	PRTCMD	Optional	PRTCMD sets the horizontal print position to column 1 after the printer receives and performs a Write and Space or Skip command.
	CR/FF			CR/FF sets the horizontal print position to column 1 after the printer receives and performs CR or FF code.
VTCODE	VT<>LF	VT<>LF	Optional	Printer performs VT function as is.
	VT=LF			Printer converts VT code to LF function; performs LF when it receives VT.
VT CH	CH 2–CH 12	CH 2 or ALL	Optional	Specifies one of channel 2 through channel 12 to be a VT channel.
	ALL			Specifies all channels (CH 2 through CH 12) as a VT channel.
DT BIT	7BIT	8BIT	Required	Specifies data bit length: 7-bit mode,
	8BIT			8-bit mode,
	7SI/SO			or 7-bit and shift-in/shift-out mode.
PARITY	NO	NO	Required	No parity.
	ODD			Odd parity.
	EVEN			Even parity.
BAUD	2400B	9600B	Required	Baud rate selection.
	4800B			
	9600B			
	19200B			
A-FULL			Required	Interface buffer almost full threshold.
	75%	75%		At 75%, the remainder of the buffer is 512 bytes.
	87.5%			At 87.5%, the remainder of the buffer is 256 bytes.
A-EMP			Required	Interface buffer almost empty threshold.
	12.5%	25%		At 12.5%, 256 bytes.
	25%			At 25%, 512 bytes.
	37.5%			At 37.5%, 768 bytes.
	50%			At 50%, 1,024 bytes.
	62.5%			At 62.5%, 1,280 bytes.

*Menu items marked with an asterisk may not appear on your model of printer.

Table 14-5 Printer Setup Menus and Fields (continued)

Menu Name	Field Names	Initial Setting	Required/ Optional	Initial Explanation
PRTCOL	DTR=RV	DTR=RV	Required	Data terminal ready (DTR) stays on until interface buffer changes from almost empty or empty to almost full state.
	RTS=RV			Request to send (RTS) line stays on until interface buffer changes from almost empty or empty to almost full state.
	XONXOF			Printer sends XON code to host on interface buffer almost empty or empty state, and sends XOFF code to host on almost full state.
	REV.ON			Reverse channel line stays on until interface buffer changes from almost empty or empty to almost full state.
	REV.OF			Reverse channel line stays off until interface buffer changes from almost empty or empty to almost full state.
STPBIT	1	1 or 2	Required	STOP BIT selection.
	1.5			
	2			
RTS	FULDPX	FULDPX	Required	The request to send (RTS) signal is always on.
	HLFDPX			The RTS signal goes on when requesting the host to transmit data and during data transmission.
				Note: This setup item is invalid under the RTS line protocol.
CTS	DISABL	DISABL	Required	Transmits data to the host regardless of whether CTS signal is on or off.
	ENABLE			Transmits data to the host only when CTS signal is on.
RI	DISABL	DISABL	Required	Disables the function that turns on the DTR signal when the ring indicator (RI) is not used by the system.
	ENABLE			Enables the function that turns on the DTR signal in response to the RI signal.
DTR	STATUS	STATUS	Required	The DTR signal depends on whether the printer is online (START state) or offline (STOP state)
	CNT ON			The DTR signal is always on. Whenever the power is on, and no error occurs, the printer is online (START state).
				This setup item is disabled under the DTR line protocol.
RLSD	DISABL	DISABL	Required	The printer accepts the received data regardless of whether the received line-signal detector (RLSD) signal is on or off.
	ENABLE			The printer accepts the received data only if the RLSD signal is on.
DSR	DISABL	DISABL	Required	The printer accepts the received data regardless of whether the data set ready (DSR) signal is on or off.
	ENABL			The printer accepts the received data only if the DSR signal is on.
SCDTXE	SUB 20–SUB 7E	SUB 3F	Required	Specifies the printing data code into which the received data is converted when a transmission error occurs. Printing data codes range from code 20 through code 7E.
SUBTXE	DISABL	DISABL	Required	DISABL does not allow received data to be converted into a substitute code when a transmission error occurs. Such data is ignored.
	ENABLE			ENABLE allows received data to be converted into the substitute code set by SCDTXE when a transmission error occurs.

Table 14-5 Printer Setup Menus and Fields (continued)

Menu Name	Field Names	Initial Setting	Required/ Optional	Initial Explanation
STPTXE	DISABL ENABLE	DISABL	Required	DISABL does not stop the printer when a transmission error occurs. ENABLE stops the printer when a transmission error occurs.
SCDIVC	SUB 20–SUB 7E	SUB 3F	Required	Specifies the printing data code into which the received data is converted when an invalid control code is received. Printing data codes range from code 20 through code 7E.
SUBIVC	DISABL ENABLE	DISABL	Required	DISABL does not allow received data to be converted into a substitute code when an invalid control code is received. Such data is ignored. ENABLE allows received data to be converted into the substitute code set by SCDIVC when an invalid control code is received.
STPIVC	DISABL ENABLE	DISABL	Required	DISABL does not stop the printer when an invalid control code is received. ENABLE stops the printer when an invalid control code is received.
SCDISQ	SUB 20–SUB 7E	SUB 3F	Required	Specifies the printing data code into which the received data is converted when an invalid control sequence is received. Printing data codes range from code 20 through code 7E.
SUBISQ	DISABL ENABLE	DISABL	Required	DISABL does not allow received data to be converted into a substitute code when an invalid control sequence is received. Such data is ignored. ENABLE allows received data to be converted into the substitute code set by SCDTXE when an invalid control sequence is received.
STPISQ	DISABL ENABLE	DISABL	Required	DISABL does not stop the printer when an invalid control sequence is received. ENABLE stops the printer when an invalid control sequence is received.
AUTONL	DISABL ENABLE	DISABL	Required	DISABL specifies that the printer remain offline (in the STOP state) when the power is turned on. ENABLE specifies that the printer is put online (into the START state) when the power is turned on.
AUTEJC	ENABLE DISABL	ENABLE	Required	ENABLE enables the following operation: If the printer is online (in the START state) and you press the TOP OF FORM button while data is neither being transmitted from the host nor stored in the interface buffer, the printer will enter the STOP state, feed four or five sheets two seconds later, and then reenter the START state after 20 seconds. DISABL disables the operation described above.
HEXDMP	DISABL ENABLE	DISABL	Required	DISABL does not allow the printer to print a hex dump of data received from the host. If hexadecimal data is transmitted, the printer stops and converts it to substitute code. ENABLE allows the printer to print a hex dump of data received from the host. The printer does not stop and try to convert data to substitute code.

ROUTINE MAINTENANCE

Routine maintenance includes:

- Loading paper
- Changing ribbons
- Replacing the print band
- Cleaning visible areas

Rational assumes responsibility for all other scheduled and unscheduled maintenance of the Rational printer.

Loading Paper

When the printer has used all its forms, the end-of-forms detector senses the absence of paper and the Status indicator displays FORMS. The printer stops operation when the current page is complete. When operation stops, follow the steps outlined below. Refer to Figure 14-4 for a diagram of the paper path for loading new forms. Refer to Figures 14-4 and 14-5 for identification of the components referred to below.

1. Press the Start/Stop switch to take the printer offline.
2. Lift the top cover.
3. Press the Top of Form switch.

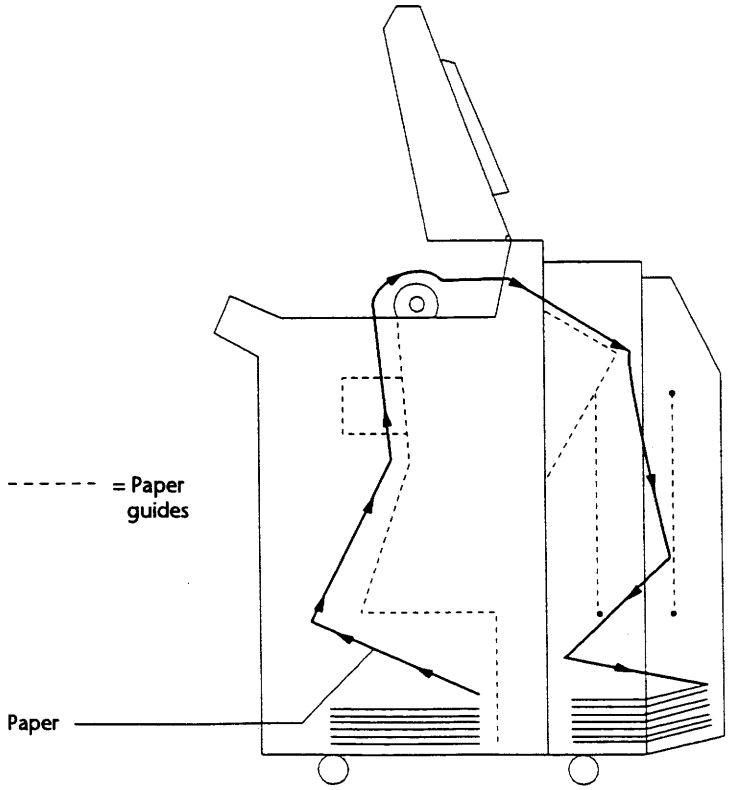

Figure 14-4 Loading Paper into the Rational Printer

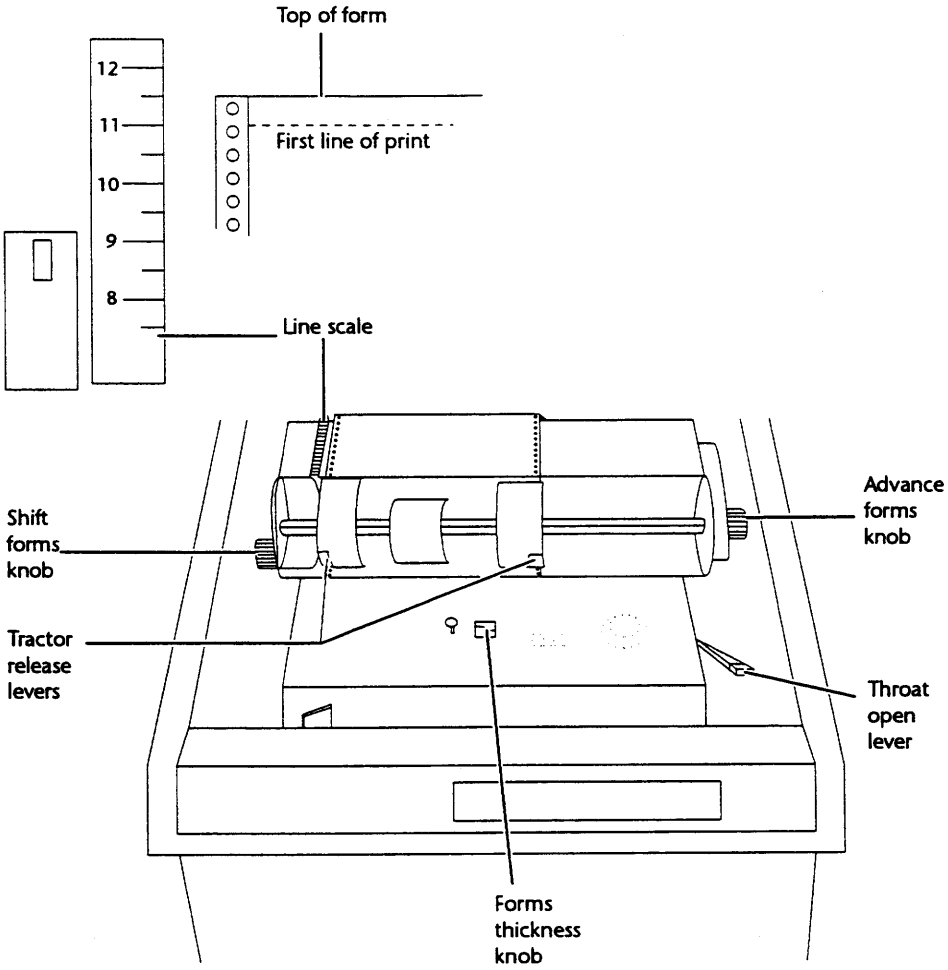

Figure 14-5 Aligning Paper in the Rational Printer

4. Lift the throat-open lever to release the print unit and increase the clearance in the forms path.
5. Open the front door.
6. Place forms in the forms hopper.
7. Put the top sheet of the forms into the forms path. It should fit between the plate and the forms guide. Push the forms upward.
8. Pull the first sheet out from the upper forms path. Uncover the forms tractors and mount the sheet on the forms tractors. Cover the forms with the tractor covers.
9. Align the top-of-forms position with the line scale on the left forms tractor. (For the location of the line scale, see Figure 14-5.) Use the advance-forms knob for initial adjustment. If necessary, use the fine-forms-adjust buttons for fine adjustments.
10. If necessary, use the tractor-release levers to adjust lateral tension in the forms.
11. If necessary, use the shift-forms knob to adjust the horizontal print position.
12. Ensure that the forms in the forms hopper are aligned so the paper can feed straight into the printer. Once the forms are aligned, pull them downward slightly to eliminate unnecessary slack.

13. If necessary, use the line scale as shown for precise forms alignment.
14. It may be necessary to use the forms-thickness knob to compensate for differences in forms thickness. Turn the knob clockwise if the forms seem too thin or counterclockwise if the forms seem too thick.
15. Close the front door, and secure the throat-open lever to lock the print unit.
16. Ensure that the forms come out of the printer correctly. When the top page reaches the bottom of the forms rack, adjust this page so that the creases in the forms align and fold correctly.
17. Close the top cover, and press the Start/Stop switch to put the printer online.

Changing Ribbon

To change the ribbon, see Figure 14-6 for the controls and components referred to in this section. See also the instructions under "Cleaning," below, for cleaning the inside of the printer after removing the ribbon cartridge.

To remove the ribbon cartridge:

1. Press the Start/Stop switch to take the printer offline.
2. Lift the top cover.
3. Lift the throat-open lever to release the print unit.
4. Pull the ribbon-drive release lever forward to increase the ribbon-feed-roller clearance.
5. Pull the handle on the left side of the ribbon cartridge and carefully remove the cartridge.

To replace the ribbon cartridge:

1. Put on plastic gloves.
2. Pull a short length of ribbon from the right side of the ribbon cartridge.
3. Hang the ribbon over the ribbon-feed-check roller. Push the cartridge into the ribbon case.

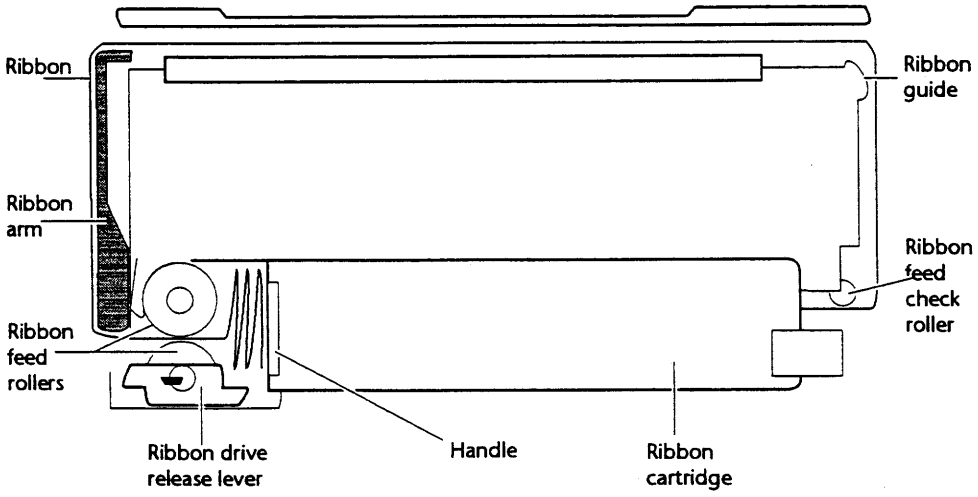

Figure 14-6 Changing the Ribbon on the Rational Printer

4. Pull a short length of ribbon from the left side of the cartridge.
   a. Pass the ribbon through the ribbon-feed rollers.
   b. Hang the ribbon over the left side ribbon arm.
   c. Pass the ribbon around the outside of the ribbon separator.
   d. Hang the ribbon over the right side ribbon guide.
5. Return the ribbon-drive release lever to its original position.
6. Turn the ribbon-feed roller by hand to reduce excess ribbon slack.
7. Push the throat-open lever down to secure the print unit.

Replacing the Print Band

See Figure 14-2 for components and controls referred to in this section.

To remove the print band:

1. Put on plastic gloves.
2. Press the Start/Stop switch to take the printer offline.
3. Lift the top cover.
4. Lift the throat-open lever to release the print unit.
5. Lift the band-cover latch on the top of the band cover and open the band cover.
6. Turn the band-exchange knob to the band-exchange position. This frees the print band by narrowing the distance between the "idle" and "drive" sides of the print-band pulley.
7. Lift the left ribbon arm to keep the ribbon raised.
8. Hold the left and right sides of the print band and carefully lift the print band upward.

   Be careful not to let the print band touch the ribbon separator, platen, or any other component.

   Be careful not to touch the print band with a magnet or screwdriver.

To replace the print band:

1. Insert the print band between the ribbon separator and platen.
2. Hang the print band on the "idle" and "drive" sides of the print-band pulley. The print band should be mounted so that the print-band rims are aligned with the upper and lower faces of the pulleys.
3. Turn the band-exchange knob to "normal."
4. Turn the "idle" pulley counterclockwise by hand until the print band is in place. This may require as many as ten rotations.
5. Use the print-band-position knob to adjust the position of the print band.
6. Close and lock the print-band cover.
7. Lower the ribbon by returning the left ribbon arm to its original position.
8. Turn the ribbon-feed roller (see Figure 14-6) by hand to reduce excess ribbon slack.
9. Close the throat-open lever.

Caution: *Do not fold or twist the print band. Do not start the print unit without locking the print-band cover.*

Cleaning

Always turn the power off before cleaning. Clean only those places that are visible to the operator.

1. When replacing the ribbon, clean the following parts after removing the ribbon cartridge:
   a. Ribbon cartridge container

      Use a vacuum cleaner and a soft cloth to remove ribbon lint from the ribbon-cartridge container and adjacent areas.

   b. Ribbon-feed rollers

      Remove ribbon lint with a soft cloth. Do not use solvents such as alcohol.

   c. Ribbon arm, ribbon guide, and ribbon-feed-check roller

      Use a soft cloth to remove lint and ink from the ribbon arm, ribbon guide, and ribbon-feed-check roller.

   d. Print band

      Remove the print-band cover. Move the print band by turning the "idle" pulley counterclockwise by hand, cleaning dirty or clogged type with gauze. Use a vacuum cleaner to remove ribbon and paper lint from the print-band area.

      Do not use solvents such as alcohol on the inside of the print band (back side of type).

   e. Tractor shaft

      Remove the forms guide. Use a soft cloth to remove dirt from the tractor shaft.

   f. Forms-feed tractors, horizontal-adjust plate, and adjacent areas

      Use a vacuum cleaner to remove dust, lint, and paper fragments.

   g. Ribbon separator

      Remove the ribbon separator. Use a vacuum cleaner to remove dust, lint, and paper fragments. Wipe off dried ink with a soft cloth moistened with alcohol or ligroin.

2. Vacuum the forms hopper periodically.
3. Clean the outside printer housing as needed.

CONFIGURING THE SYSTEM'S PRINTER PORT

For the Series 200, the conventional printer port is terminal port 31 on both 16- and 32-port systems. For the Series 200, 300, and 400, printers are also typically connected to a network.

The preferred method for configuring printer ports is through procedures in !Machine.Initialization. See Appendix E for more information.

The port settings can also be changed explicitly by using Environment commands (see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*, packages Terminal, Operator, and Queue). You will not need to change the standard port settings, however, unless you connect the Rational printer remotely, connect a different printer to the R1000, or move the printer from one port to another.

Note: *When the printer is connected to a 20/40 printer switch, the printer-setup-mode settings must match the printer-port settings for both R1000 processors.*

The standard port configuration for a Series 200 assumes that a Rational printer is connected directly to the R1000.

Changing Printer Port Settings

The port settings can be changed using Environment commands. However, you will not need to change the standard port settings unless you connect the Rational printer remotely, connect a different printer to the R1000, or move the printer from one port to another.

The commands that control port settings are in package !Commands.Terminal. For example, the Commands.Terminal.Set_Input_Rate command sets the input baud rate, and the Commands.Terminal.Set_Character command sets the character size.

Note that, although enabling the print queue sets the printer-port flow control for transmit data to the correct setting, this setting can be overwritten by Terminal-.Set_Flow_Control. However, the change may be very temporary. No matter what value you assign, enabling the queue resets the flow-control value.

Port Configuration and Printer Setup

Most of the printer-port settings (all, except terminal type, Xon, and Xoff) correspond to setup menu items on the printer itself. (The setup menu items are explained in the "Setup Mode" section earlier in this chapter.) The printer-port settings must always match the corresponding printer-setup-mode settings. Therefore, if you change a port setting using a command from package !Commands.Terminal, you also must verify that the corresponding setting on the printer matches the new port setting. If it does not, you need to change either the port setting or the printer setting to achieve the required match.

Note: *When the printer is connected to a 20/40 printer switch, the printer-setup-mode settings must match the printer-port settings for both R1000 processors.*

Observe in Table 14-6 that the port settings in the standard configuration listed above match the corresponding printer settings listed in Table 14-5, "Printer Setup Menus and Fields," earlier in this chapter.

Table 14-6 Rational Printer and R1000 Port-Setting Correspondences

Port Parameter	Value	Printer-Setup Menu Item	Printer-Setup Field Item
Input baud rate	BAUD_9600	BAUD	9600B
Output baud rate	BAUD_9600	BAUD	9600B
Parity	NONE	PARITY	NO
Stop_Bits	2	STPBIT	2
Char_Size	CHAR_8	DT BIT	8BIT
Flow control for transmit data	NONE	PRTCOL	DTR=RV

SETTING UP AND MANAGING THE PRINT SPOOLER

Whenever the system boots, the print spooler is normally started by software in !Machine.Initialization. You can manually execute this process by specifying "Printers" as the argument to the Procedures_To_Run parameter and False as the argument to the Effort_Only parameter—for example:

```
Start (Rational_Initialization_Directory =>
                                "!Machine.Initialization.Rational",
       Site_Initialization_Directory     =>
                                "!Machine.Initialization.Site",
       Local_Initialization_Directory    =>
                                "!Machine.Initialization.Local",
       Procedures_To_Run                 => "Printers",
       Effort_Only                       => False);
```

You can also make custom changes to the Printer_Configuration file used by machine initialization. See Appendix E for more information.

Although the preferred method for configuring the print spooler is to use the facilities in machine initialization, an alternative is to make temporary changes through the commands in package Queue. See package Queue in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* for complete instructions.

Making Changes through Commands in Package Queue

As an alternative to using the facilities available through machine initialization, you can make temporary changes through the commands in package Queue. Refer to package Queue in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* for complete instructions.

If you want to change the printer-port settings, you must disable the port first. For example, you can disable printer port 31 by executing the command:

```
Queue.Disable("Terminal_31")
```

Note: *You will not be able to disable the port if the printer is in use.*

When the printer is connected to a 20/40 printer switch and you want to disable the printer, you must disable the printer ports on both CPUs. If you change parameters, you must change the parameters to be identical for both ports.

To reenable the disabled port, enter the command:

```
Queue.Enable ("Terminal_31")
```

Example 1: Changing the Flow Control

Instead of connecting the printer directly to the R1000, you may want to connect it through a switch or controller. If the printer is connected through a switch that does not accept hardware flow control (DTR protocol), you must change the printer-port configuration and the printer settings to use software flow control. Note that you cannot use software flow control for the printer switch. In the following, assume that the printer port is 31.

1. Make sure the printer is not in use, and then disable port 31:

   ```
   (Queue.Disable ("Terminal_31"))
   ```

2. Enable software flow control with the following command:

```
(Terminal.Set_Flow_Control (31, True))
```

3. If desired, specify nondefault Xon and Xoff characters. (The defaults are DC1 and DC3, respectively.) Use the Terminal.Set_Xon_Xoff_Characters command to specify characters. Use the Terminal.Set_Xon_Xoff_Bytes command to specify bytes. (Use bytes if you need the whole character set for other purposes.)
4. Enter setup mode on the printer and display the PRTCOL menu item. Change its field item to XONXOF.
5. Enable port 31 with the following command:

```
Queue.Enable ("Terminal_31"))
```

6. Take a snapshot to preserve the new port configuration. (For the procedure, see "Taking a Snapshot by Explicit Request" in Chapter 5.)

Example 2: Moving the Printer to Another Port

If you need the printer port for another purpose, you can move the printer to another port. In the following, assume that the printer is to be moved from port 31 to port 47.

1. Make sure that ports 31 and 47 are not in use, and then disable them:

```
Queue.Disable ("Terminal_31");
Queue.Disable ("Terminal_47");
```

2. Check the settings on port 47 to make sure they are compatible with the printer. Use the following command to display the port settings:

```
Terminal.Settings (47);
```

3. Change settings on port 47, as needed, using the appropriate command from package !Commands.Terminal.
4. Remove the old port from the print spooler:

```
Queue.Remove ("Terminal_31", True);)
```

5. Register the new port to the print spooler, and establish its class—for example, LP:

```
Queue.Register ("Terminal_47", "LP");
```

6. Enable the new port:

```
Queue.Enable ("Terminal_47");
```

15 Tape Drives

This chapter, describing the operation and maintenance of tape drives, is divided into three parts:

- 9-track tape drives
- 8-mm tape drives
- Dual drives on the Series 200

9-TRACK TAPE DRIVES

This section includes information on 9-track tape-drive operations, diagnostics and error-recovery procedures, and routine maintenance procedures. Rational support personnel assume responsibility for all other tape-drive maintenance, whether scheduled or unscheduled. If any problems occur, contact your Rational support representative using the problem-reporting procedures described in Appendix D.

The *Rational Environment Reference Manual* contains information on procedures for reading from and writing onto tape. See the System Management Utilities (SMU) and Library Management (LM) books, packages !Commands.Tape and !Commands.Archive, respectively.

Kennedy Drive (Series 200, Model 10, and Series 300)

This subsection describes tape-drive operations and maintenance for the Kennedy 9-track tape drive mounted on the Rational R1000 Series 200, Model 10, and the Series 300.

Tape-Drive Control Panel

The Kennedy tape-drive control panel on the R1000 includes eight push-button switches for controlling tape-drive functions and an eight-character LED display panel that displays operating mode, status, parameters, and diagnostic messages (see Figure 15-1). There are also indicators mounted in several of the switches. To the left of the control panel is the power on/off switch.

The power switch controls the AC main power to the unit. Power is applied when the upper end (1) of the switch is pressed and removed when the lower end (0) of the switch is pressed. When the power is switched on, the character display shows the message `TESTING` for about 15 seconds. If this power-up test completes without detecting an error, the character display shows the message `UNIT 0`.

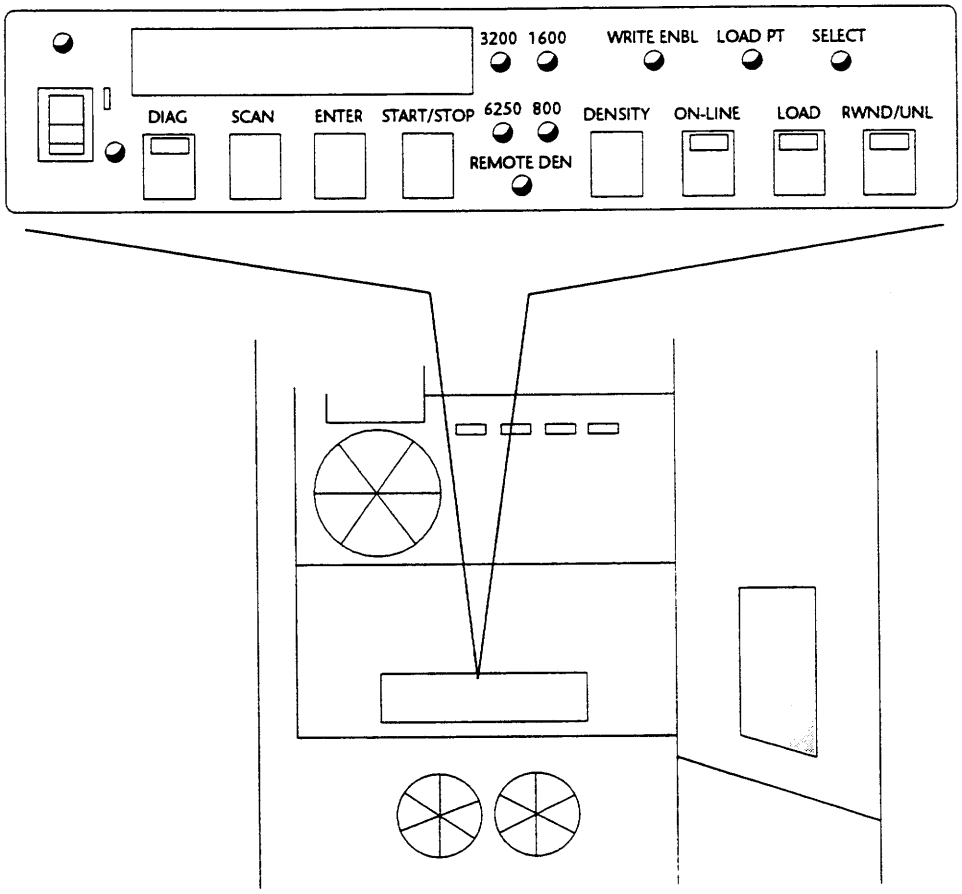

Figure 15-1 Model 10: Tape-Drive Control Panel

Table 15-1 describes the switches and indicators on the control panel.

Table 15-1 Model 10: Tape-Drive Control Panel

Switch or Indicator	Function
1/0	(Switch and Indicator) Controls AC power to the unit. The switch lights internally when power is switched on.
Diag	(Switch and Indicator) Pressing this switch turns diagnostics on and off when the unit is offline. The indicator lights when the unit is in diagnostics mode.
Scan	(Switch) Scans through the diagnostics and calibrate menus.
Enter	(Switch) Enters selected commands when in diagnostics or calibrate mode.
Start/Stop	(Switch) Starts or stops selected commands while in diagnostics or calibrate mode.
Remote den	(Indicator) Illuminates when density is selectable by the host system.
800/1600/ 3200/6250	(Indicators) Indicates which density has been selected.
Density	(Switch) Selects the desired density mode of the unit. Pressing this switch cycles through the four densities and alternates between remote and local modes.

Table 15-1 Model 10: Tape-Drive Control Panel (continued)

Switch or Indicator	Function
Online	(Switch and Indicator) Places the unit online with the host system. The indicator illuminates when the unit is online.
Load	(Switch and Indicator) Pressing this switch initiates a load sequence. The indicator illuminates when the load sequence has successfully completed.
Rwnd/Unl	(Switch and Indicator) Initiates a rewind or unload operation when the unit is offline. The indicator illuminates when a rewind or unload operation is in progress.
Write Enbl	(Indicator) Illuminates when the write-enable ring is not installed on the supply reel.
Load Pt	(Indicator) Illuminates when the tape is positioned at the beginning-of-tape (BOT) marker.
Select	(Indicator) Illuminates when the unit is selected by the host system.

Mounting a Tape

Note: *This tape unit features an automatic load process. The entire load process takes 30–45 seconds after the Load switch has been pressed.*

To load a tape, refer to Figure 15-2 while performing the following three steps:

1. Open the tape-access door by pressing on the lower portion of the door.
2. Position the supply reel so the write-enable-ring side is down. Insert the supply reel as shown in the drawing, mounting the supply reel over the supply-reel hub.

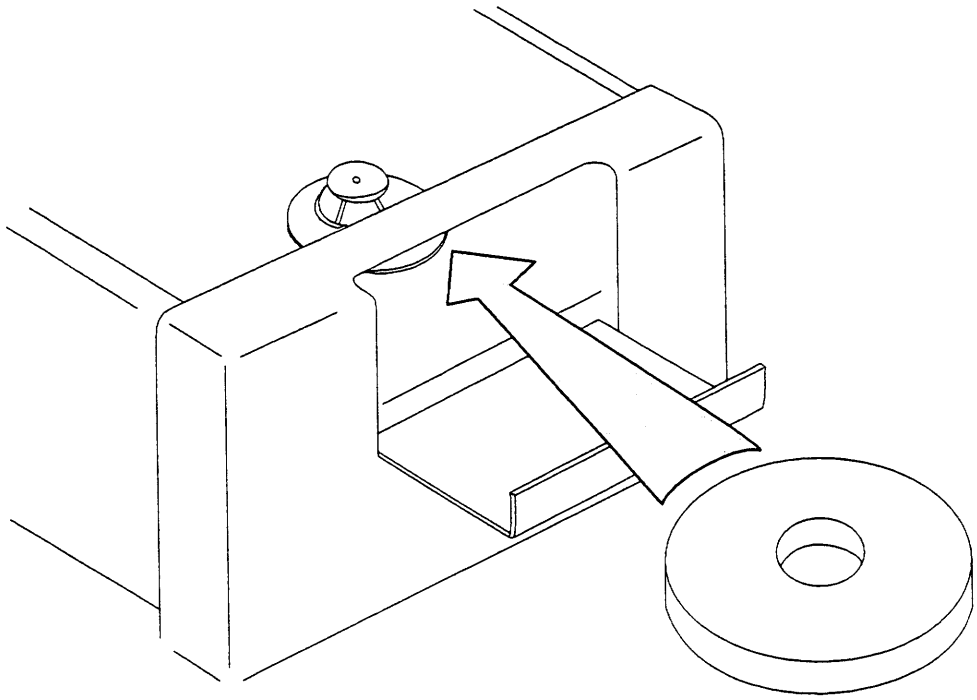

Figure 15-2 Model 10: Mounting a Tape

3. Close the tape-access door and press the Load switch on the control panel. Within approximately 45 seconds, the tape is threaded automatically through the unit's tape path, with the tape positioned at the beginning-of-tape (BOT) marker. At this point, the Load Pt indicator illuminates.

If the unit is unable to load the tape, the display panel displays a diagnostic message. Refer to Table 15-2 for some of the common error codes.

If the load process fails for one of the reasons listed in Table 15-2, correct the condition and repeat the tape-loading operation.

If the error encountered is not a load failure, contact your Rational support representative with information relevant to the problem.

Table 15-2 Model 10: Common Tape-Drive Error Codes

Error Code	Explanation
CLOSE/DOOR	Tape access door is open.
BOT	Tape marker is missing from tape.
REV REEL	Tape reel is upside down.
BKN TAPE	Tape is broken or torn.

Setting Tape Density and Local/Remote Density Control

By repeatedly pressing the Density switch, you can set the 9-track tape drive for 1600 BPI or 6250 BPI density and for local or remote control.

In read operations, if the drive is in remote mode, the drive automatically reads in the density in which the tape was written.

In write operations, the drive writes in the density indicated by the Density indicators.

Unloading a Tape

To unload a tape from the take-up reel and rewind it onto the supply reel:

1. If the unit is online, press the Online switch to take the unit offline.
2. Press the Rwnd/Unl switch. The entire tape rewinds onto the supply reel. As the operation proceeds, the display panel displays the message UNLOADING. When the operation has completed, the display panel displays the message UNIT 0.
3. When the tape-access door unlocks, remove the tape from the unit.

Diagnostics

Diagnostic operations occur under three conditions:

- Power-on diagnostics are performed routinely each time the tape drive is turned on.
- Online diagnostics are initiated by the Environment.
- Offline diagnostics are built into the tape drive itself and are invoked by the operator or Rational's support representatives.

During operation, the tape drive tests for error conditions and displays an error message in the eight-character display panel. See Table 15-2 for a list of some common error codes and their meanings.

When an error condition occurs:

1. Interpret the fault code. If the error is not one of the common errors encountered while loading tape, be sure that the regular daily operator maintenance has been performed and that the fault is not due to a poor-quality tape (try using a new scratch tape).
2. If the cause of the problem is not determined in step 1, execute the following self-test. (This self-test can be aborted at any time by pressing the Diag button.)
   a. Place a write-enabled 2,400-foot tape into the tape drive, but do not load it.
   b. Activate Diagnostics mode by pressing the Diag switch. `SELFTEST` is displayed.
   c. Enter the self-test by pressing the Enter switch. `PWUPTEST` is displayed.
   d. Press the Scan switch until `SELFTEST` is displayed.
   e. Press the Enter switch. `CONFIRM?` is displayed.
   f. Start the test by pressing the Online button.
3. If the self-test terminates with any message other than `PASSED`, record this message and contact your Rational support representative.

Cleaning Procedures

Caution: *Never clean the unit with power on. If, with power on, the takeup arm is inadvertently moved past the limit sensor, the drive-arm motor drives the arm into the chassis wall, destroying calibration and the drive's ability to load (see Figure 15-3).*

Clean the following areas regularly—approximately every eight hours of tape motion. Refer to Figure 15-3 for the location of the components listed below:

- Tape path
- Magnetic read/write (R/W) head (under tape-path cover)
- Capstan (under tape-path cover)
- BOT/EOT sensors (under tape-path cover)
- Rollers
- Supply-reel hub
- Tape-access door

To gain access to some of these components:

1. Turn the power off.
2. Pull out the drive.
3. Lift the inside edge of the tape-path cover.

Cleaning Materials

You will need the following cleaning materials for regular operator maintenance:

- Lint-free cloth
- Foam swabs
- Tape transport cleaner or isopropyl alcohol

Caution: *Do not use tape transport cleaner in a confined or poorly ventilated area. Avoid breathing the vapor. Use disposable gloves to avoid prolonged contact with skin. Read and follow all precautions listed on any cleaning-solvent container.*

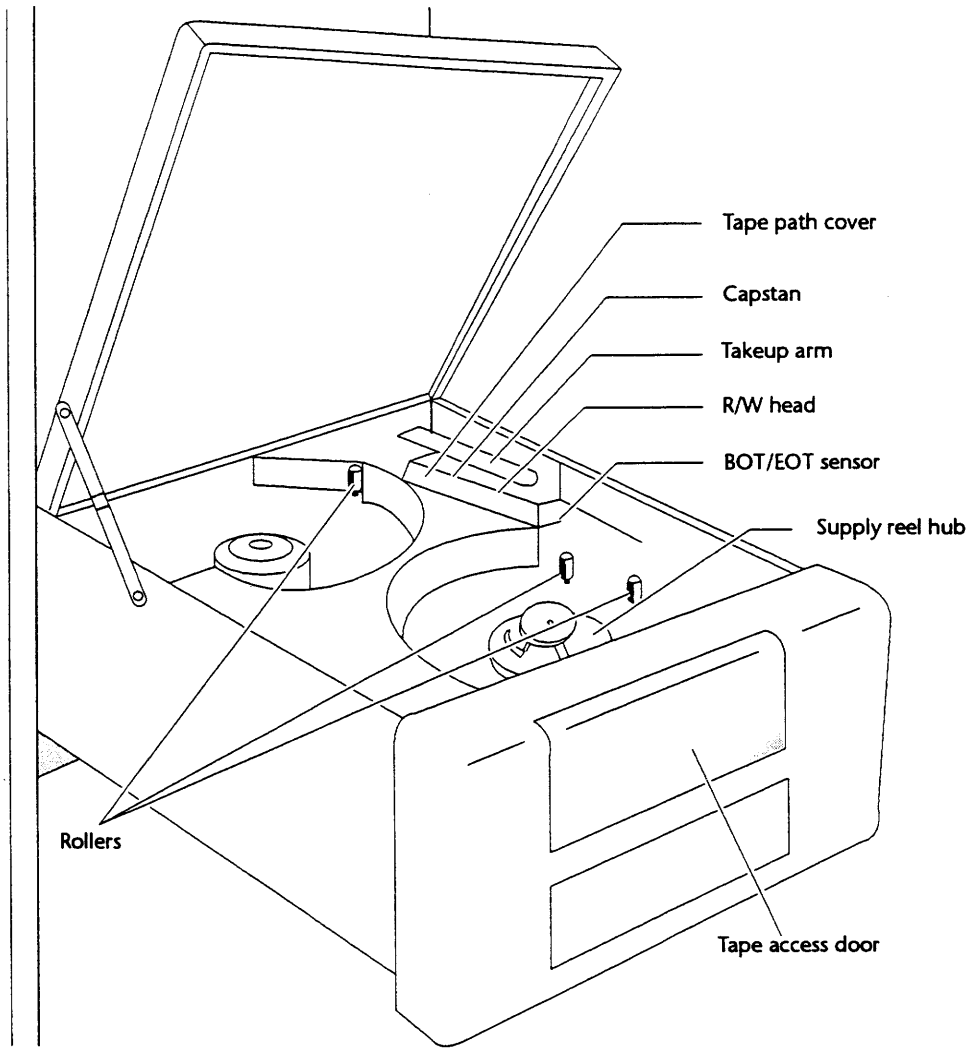

Figure 15-3 Model 10: Tape-Drive Components for Cleaning

Magnetic Head

Remove any accumulation of oxide and/or dirt from the surface of the erase/write/read head using a clean, lint-free cloth or foam swab dampened with isopropyl alcohol. See Figure 15-3.

Tape Path

Clean the tape guides and rollers with a foam swab dampened with tape transport cleaner or isopropyl alcohol. Clean the capstan rubber roller with a foam swab dampened with *water* only.

Fujitsu Tape Drive (Series 200, Models 20 and 40)

This subsection describes tape-drive operations and maintenance on the 9-track tape drive mounted on the Rational R1000 Series 200, Model 20 and Model 40.

Tape-Drive Control Panel

The 9-track tape-drive control panel for the R1000 Series 200, Models 20 and 40, includes touch-sensitive switches and a two-digit display for error codes as shown in Figure 15-4. There also are indicators mounted in several of the switches. The power on/off switch is mounted to the right of the control panel.

The power switch controls the AC main power to the unit. Power is applied when the 1 (upper end) of the switch is pressed. Power is removed when the 0 (lower end) of the switch is pressed. When the power is turned on, all indicators for the control panel are illuminated for one second during the indicator malfunction test. Power-on diagnostics are automatically executed after all DC voltage lines stabilize, and then the two-digit indicator on the control panel indicates a normal code, `00`.

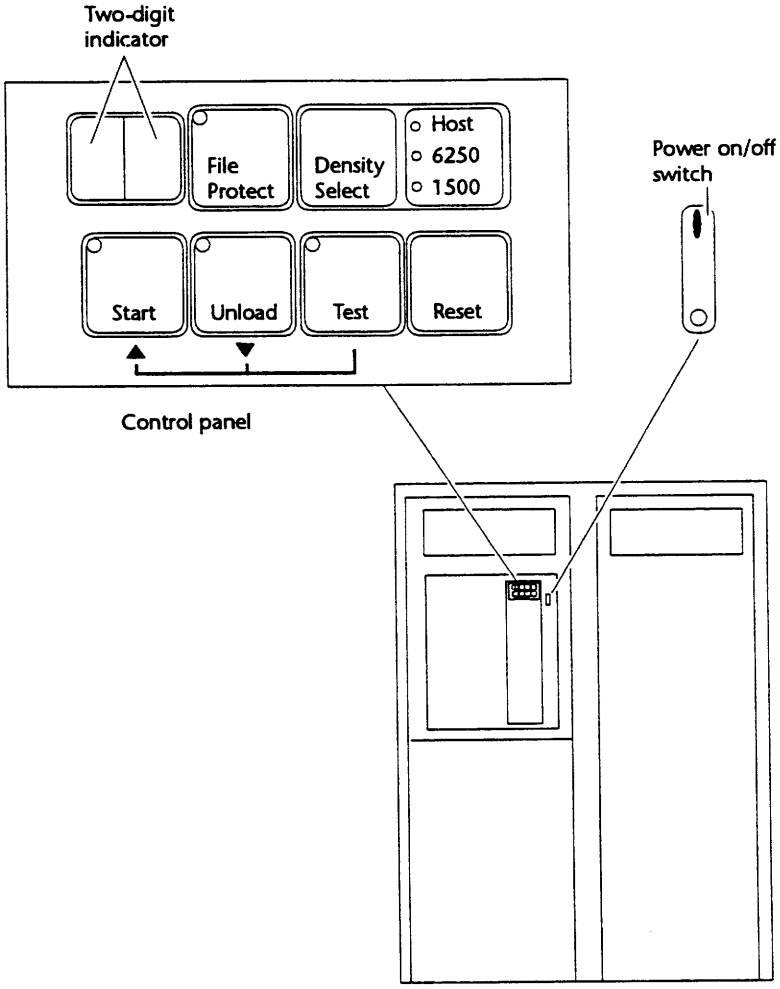

Figure 15-4 Models 20 and 40: Tape-Drive Control Panel

Table 15-3 describes the switches and indicators on the control panel.

Table 15-3 Models 20 and 40: Tape-Drive Control Panel

Switch or Indicator	Function
(Two-digit indicator)	Indicates the power-on state or the details of unit errors. The normal power-on state is indicated by a `00` display. This indicator displays `LP` when the tape is initially loaded and positioned at the beginning-of-tape (BOT) marker.
File protect	(Indicator) Illuminates when the write-enable ring is not mounted on the supply reel. A write operation is inhibited if this indicator is illuminated.
Density select	(Switch) Selects the desired density mode of the unit. Pressing this switch cycles through to 6,250 (GCR) or 1,600 (PE), or it places the unit into the mode of allowing the host system to select one of the two densities. This switch is ignored when the unit is online or in diagnostic mode.
Host/6250/ 1600	(Indicators) Indicate the density mode while writing to tape. `Host` indicates the unit can change the write density by a command from the host; `6250` indicates that the write density is set to 6,250 (GCR); `1600` indicates that the write density of the unit is set to 1,600 (PE).
Start	(Switch and Indicator) When tension is not applied to the tape, this switch initiates the load sequence. After the load sequence is completed, the unit becomes ready and the indicator illuminates.
Unload	(Switch and Indicator) This switch is effective only when the unit is not in the ready state. Pressing this key rewinds the entire tape onto the supply reel. The indicator is on during the unload operation.
Test	(Switch and Indicator) Invokes the unit's diagnostic tests.
Reset	(Switch) Places the unit in the not-ready state.

Mounting a Tape

To mount a tape, refer to Figure 15-5 while performing the following steps:

1. Open the front door.
2. Remove or install the write-enable ring on the supply reel, as required.
3. Position the supply reel so the write-enable ring is on the side of the reel facing the drive. Mount the supply reel over the supply-reel hub.
4. Thread the beginning of the tape over the tension roller and through the upper tape guide. Place the tape between the R/W head and the head shield and pull the tape through the lower tape guide. Wrap the tape around the idler roller and onto the unit's take-up reel.
5. Close the front door and press the Start switch on the control panel. The tape will begin forward motion, stopping at the beginning-of-tape (BOT) marker.

Setting Tape Density

To select the tape density, press the Density Select switch to cycle through the available tape densities: 6,250 (GCR), 1,600 (PE), or Host Select (which allows the host system to select one of the two densities). If you press this switch while the unit is online or in diagnostic mode, it is ignored.

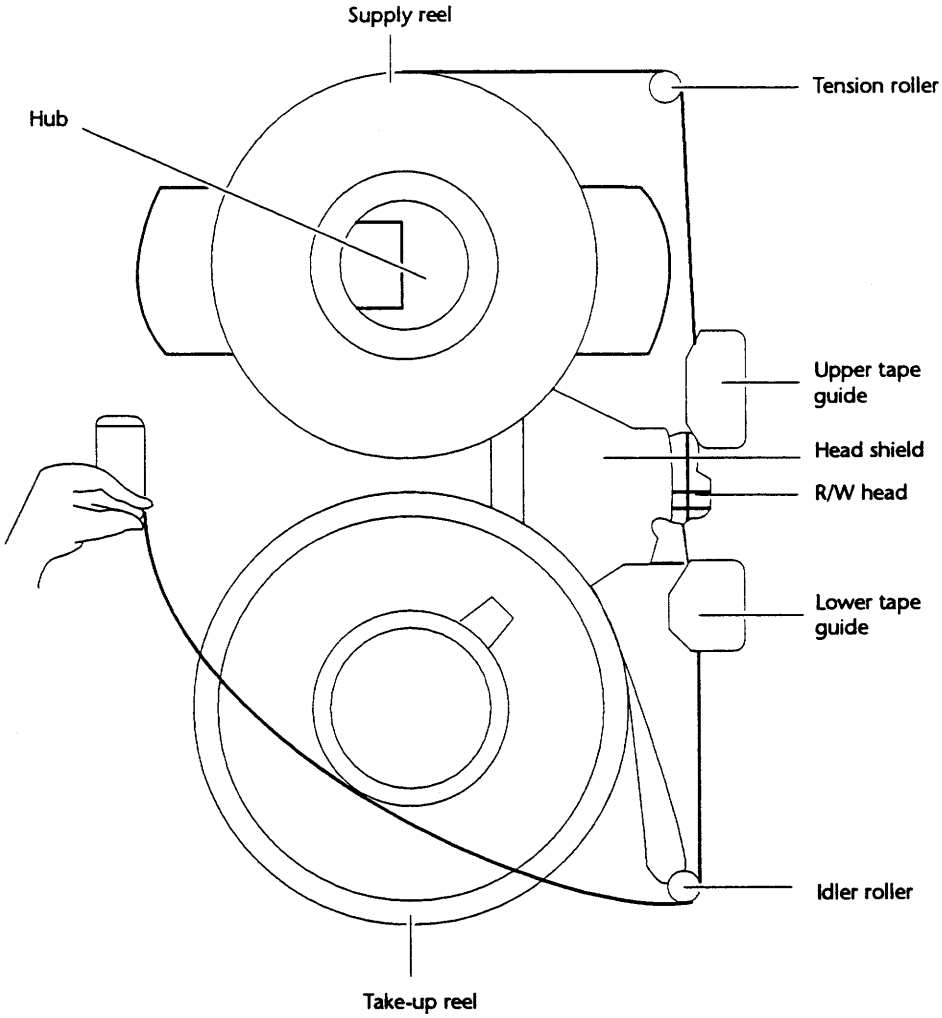

Figure 15-5 Models 20 and 40: Mounting a Tape

Unloading a Tape

To unload a tape from the take-up reel and rewind it onto the supply reel:

1. Press the Reset switch on the control panel to take the tape drive offline.
2. Press the Unload switch. The entire tape rewinds onto the supply reel.

Diagnostics

Diagnostic operations occur under three conditions:

- Power-on diagnostics are routinely performed each time the tape drive is turned on.
- Online diagnostics are initiated by the Environment.
- Offline diagnostics are built into the tape drive itself and are invoked by the operator or Rational's support representatives.

During operation, the tape drive tests for error conditions and displays an error code in the two-digit indicator. See Table 15-4 for a list of error codes and their meanings.

Table 15-4 Models 20 and 40: Tape-Drive Error Codes

Error Code	Explanation
L1	Door is opened when servo is on or when the Start or Unload switch is pressed.
L2	Reel latch is unlocked.
L3	Tape is loose.
L4	Beginning-of-tape (BOT) marker is not found.
L5	Tape is not properly positioned in the tape path.

When an error condition occurs:

1. Interpret the fault code. Refer to the label on the inside of the front door for error codes L1 through L5, possible causes, and recommended action.
2. If the error encountered is not a load failure (L1 through L5 code), execute the 01 diagnostic test by performing the following steps:
   a. Install the write-enable ring on a scratch tape. Mount and thread the scratch tape on the drive.
   b. With the tape threaded but not loaded, press the Start switch while pressing the Test switch.
   c. When the Density Select mode indicator switches to Host, press the Start switch to increase the `00` display to `01`. If the two-digit display increases beyond `01`, it can be decreased by pressing the Unload switch.
   d. Press the Test button to begin the test.
3. If the `01` diagnostic terminates with any indication other than `00`, record the numbers on the display and contact your Rational support representative with information relevant to the problem.

Cleaning Procedures

Clean the following areas approximately every eight hours of use time. Refer to Figure 15-5 for the location of the components listed below:

- Magnetic read/write (R/W) head
- Supply-reel hub
- Tension roller
- Idler roller
- Housing
- Front door

Cleaning Materials

You will need the following cleaning materials for regular operator maintenance:

- Lint-free cloth
- Foam swabs
- Tape transport cleaner or isopropyl alcohol

Caution: *Do not use tape transport cleaner in a confined or poorly ventilated area. Avoid breathing the vapor. Use disposable gloves to avoid prolonged contact with skin. Read and follow all precautions listed on any cleaning-solvent container.*

Magnetic Head

Clean the head recording surface with a lint-free cloth moistened with tape transport cleaner. Wipe the recording surface in the direction in which the tape moves across the head.

Proper Care of 9-Track Tapes

Use top-quality tapes that are suitable for GCR recording. A tape should be retired after 10 to 20 uses. To prevent the accumulation of unwanted particles, store tapes with their covers installed in a clean environment. The temperature and humidity of the tape-storage area should conform to the recommendations of the tape manufacturer.

Tape errors typically arise because of "bubbles" between windings of tape or creases in the tape. To prevent movement of the tape on the reel during handling and storage (the cause of bubbles), store tapes in a vibration-free place and with tensioners securely installed. (Tensioners are either plastic strips or rubber plugs.) When mounting tapes, check carefully for bubbles. Remove any bubbles by carefully unwinding and rewinding the tape. At all times, avoid creasing the tape.

8-MM TAPE DRIVES

This section includes information on 8-mm tape-drive operations, diagnostics, and error-recovery procedures, and routine maintenance procedures. Rational support personnel assume responsibility for all other tape-drive maintenance, whether scheduled or unscheduled.

The *Rational Environment Reference Manual* contains information on procedures for reading from and writing to tape. See package !Commands.Tape in the System Management Utilities (SMU) book and package !Commands.Archive in the Library Management (LM) book for additional information.

When an 8-mm tape drive is present, it is mounted on the right side of the front of the CPU bay (see Figure 15-6). The front of the tape drive consists of a tape-access door, an unload button, and two indicator LEDs, as shown in Figure 15-6.

A green LED is located to the right of an amber LED. The green LED lights up after the tape cartridge is loaded and ready. The amber LED blinks variably during normal tape operations, indicating I/O activity between the tape drive and the CPU.

Front-Panel Status LEDs

Table 15-5 identifies and describes the status LEDs associated with the 8-mm tape drive.

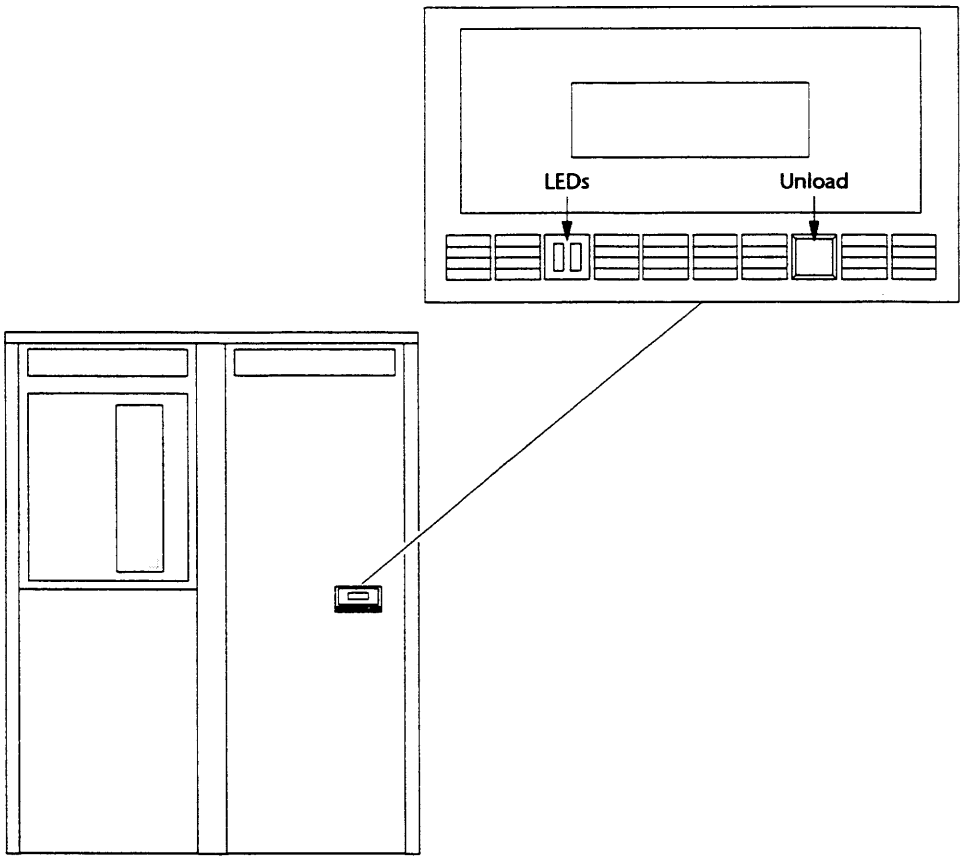

Figure 15-6 8-mm Tape Drive Mounted in a Model 20

Table 15-5 Front-Panel Status LEDs For 8-mm Tape Drive

LED	Explanation
No LEDs	Either: ■ There is no tape in the drive ■ The tape-drive door is open ■ The drive mechanism has not yet loaded the tape
Both LEDs flickering	A tape operation is in progress
Both LEDs steady	Tape drive is hung
Green LED only	Tape is loaded
Amber LED only	Hardware servo error

Note: *See "Resetting the Drive After a Servo Error," below, if only the amber LED is illuminated.*

Power-On Diagnostics

The 8-mm tape drive performs self-diagnostics each time the R1000 is powered on. During power-on initialization, the amber and green LEDs are both on, indicating

that the tape drive is performing power-on self-test diagnostics. The time required to complete self-test diagnostics and initialization routines is 120 seconds maximum. When the diagnostics are complete and the tape drive passes its self-diagnostics, both LEDs go off. If the self-test fails, both LEDs flash.

If your tape drive fails during power-up, contact your Rational support representative using the problem-reporting procedures described in Appendix D.

Choosing a Tape Cartridge

Rational recommends you use either EXATAPE™ 8-mm data cartridges or Sony® Video 8 8-mm tape.

Tape sizes are indicated in several ways, depending on the brand of tape. EXATAPE tapes are identified by their length in meters. Sony tapes are identified by their cartridge type, which indicates minutes of playing time (if they are used as video tapes). For example, P6-15MP and P5-15MP indicate 15-minute tapes. P6 mode tapes, which are used in North America, have a smaller data capacity in megabytes than P5 mode tapes, which are used in Europe. Both P5 and P6 tapes are listed in Table 15-6, which shows the amount of data, in megabytes, that will fit on a single tape cartridge.

Table 15-6 Capacity of 8-mm Tapes

Kind of Tape	Tape Capacity in Megabytes
Sony P6-15MP Video 8 Sony P5-15MP Video 8	256
EXATAPE 8-mm Data Cartridge 15m	300
Sony P6-30MP Video 8 Sony P5-30MP Video 8	512
Sony P6-60MP Video 8 Sony P5-60MP Video 8	1,024
EXATAPE 8-mm Data Cartridge 54m	1,200
Sony P6-90MP Video 8 Sony P5-60MP Video 8	1,648
Sony P6-120MP Video 8 Sony P5-90MP Video 8	2,048
EXATAPE 8-mm Data Cartridge 112m	2,500

Tape Density

The 8-mm tape drive reads and writes tapes at a fixed density; it is not possible or necessary to set the tape density manually.

If you are transferring information from another machine to the R1000, make sure the data was written using the same model of 8-mm tape drive as on the R1000 (an EXB-8200 CTS).

An 8-mm tape cartridge written by another R1000 machine should be readable on your R1000.

Mounting and Dismounting a Tape Cartridge

Use the following procedures to mount and dismount a tape.

Mounting a Tape

1. If the tape-drive door is not already open, open it by pressing the Unload button located on the front of the drive.

   If you are mounting a tape for a write operation, make sure your tape cartridge is write-enabled. If the tape is still write-protected, the write-protect tab, located on the right front edge of the cartridge, will be visible. Use a suitable instrument to slide the write-protect tab over so it is no longer visible.
2. Slide the tape cartridge into the tape drive. The cartridge should be facing up so you can read the brand-name label.
3. Close the tape-drive door.

   After about 30 seconds, the green LED should illuminate.

The tape drive is now ready to read or write the tape.

Resetting the Drive After a Servo Error

Servo errors, which are indicated when only the amber light is illuminated, sometimes occur when loading a new tape. If a servo error occurs:

1. Press and hold the tape drive's Unload button for 3–5 seconds.
2. Remove the tape.
3. Load a scratch tape from which you do not mind losing data.
4. After the green light illuminates, press and hold the tape drive's Unload button for 3–5 seconds.
5. Remove the scratch tape.
6. Reload your desired tape.

Caution: *Following a servo error, data may be lost from the next tape loaded. Be sure to load a scratch tape before the desired tape when recovering from a servo error.*

Removing a Tape:

1. Make sure the tape operation has completed. If the green or amber LEDs are blinking, the tape operation has not completed.

   Caution: *You should never try to remove the tape from the tape drive before the tape operation is complete. Doing so causes the tape operation to fail.*

   In most cases, after the tape-drive operation has completed, the tape-drive door opens automatically. (Note that you may have to wait approximately 60 seconds before the door opens.) If the tape-drive door does not open after the tape operation has completed, open the door by pressing the Unload button.
2. Pull the tape cartridge out of the drive. If you are not planning to mount another tape for some time, you should close the tape-drive door to protect the interior of the tape drive from dust.

Releasing a Tape That Is Stuck in the Drive

1. Press the Unload button on the front of the tape drive.
2. Execute the command:

```
Tape.Rewind
```

3. Then execute the command:

```
Tape.Unload
```

If the above steps fail, contact your Rational support representative for assistance (see Appendix D).

Tape-Cartridge Reliability

Use the following procedures to ensure maximum tape reliability:

- Use only the EXABYTE® tape cartridge or the Sony Video 8 metal particle, data-grade tape cartridge.
- Maintain your tape-storage environment within a range of 17°C (62°F) to 20°C (68°F), with an ideal being 18.3°C (65°F).
- Store tapes where the relative humidity is 20%–80%, noncondensing.
- Store tapes in a clean environment, with their covers installed, to prevent the accumulation of unwanted particles.
- Use only approved cleaning tapes: EXATAPE, 3-pass (part number 727113) or EXATAPE, 12-pass (part number 727114).
- Avoid using faulty or damaged cartridges.

Tape-Drive Maintenance

Rational recommends that you clean the tape head and path after *30 gigabytes of data transfer, or once a month*, whichever comes first.

Rational includes a blank 8-mm tape cartridge and a three-pass EXABYTE cartridge-cleaning kit with every 8-mm tape drive. The cleaning kit contains instructions and all items required to clean the tape drive. No other cleaning procedure or product is approved for use with the 8-mm tape drive (except a 12-pass EXATAPE cleaning tape). Other cleaning materials and tapes, including Sony dry-cleaning tapes (PN V8-25CL), cloth swabs, cotton swabs, or cleaning agents other than Freon TF reagent grade are not accepted for use and *will void the warranty on the EXB-8200 CTS.*

Resetting the Tape Drive If a Job Is in Tape_Wait

Suppose you find that a job that accesses the tape drive, such as a backup or an archive, does not finish; when you check the executing jobs through the following command:

```
====>> Elaborator Database <<====
Kernel: show_task_states
```

you find that the state for your job is Tape_Wait.

Your job is in an infinite loop, attempting to reach the tape drive, but the tape drive is not responding.

Caution: *Do NOT kill the job because this action will hang your system.*

Check for your job's state by executing Show_Task_States from the `Kernel:` prompt. If you see a job in Tape_Wait, you can use the Job_Name command from the EEDB kernel to get the name of the job. The VPID number required by this command is obtained by taking the last two digits of the task ID and converting it from hexadecimal to decimal.

For example:

```
Kernel: show_task_states
CACHE/DISK [CACHE]:
16#20CF6#; IN_WAIT_SERVICE TCP_IP_INPUT_WAIT; PRI 14
16#4238804#; IN_WAIT_SERVICE X25_WAIT; PRI 1
16#3CCF4#; UNBLOCKED; PRI 3
16#DE49404#; HELD_BY_MTS; PRI 3
16#4B0F1#; IN_WAIT_SERVICE TAPE_WAIT; PRI 14
16#4D0E2#; HELD_BY_MTS; PRI 3
16#DDE3804#; DELAYING_IN_WAIT_SERVICE U031; PRI 3
16#6D004#; UNBLOCKED; PRI 1
Kernel:
```

The following job is in Tape_Wait:

```
16#4B0F1#; IN_WAIT_SERVICE TAPE_WAIT; PRI 14
```

To find out what job this is, examine the task ID: `16#4B0F1`. Convert the last two hexadecimal digits, F1, to decimal 226. Then enter 226 as the parameter to the Job_Name command:

```
====>> Elaborator Database <<====
EEDB: kernel
Kernel: job_name
VPID [16]: 226
 Job CPU%     Root   Job Seg Acts   Name
---- ---- -------- --------- ----  -------------------------
 226    0        0   3097D10    3 !USERS.RH % ARCHIVE.SAVE
Kernel: quit
EEDB:
```

On a model 400, from the operator [Break] key menu, run the option to reset the tape drive. On other models, run the command:

```
Fix_The_8mm_Drive
```

to reset the tape drive. If this does not work, your only option is to recycle power to the drive:

- On the Series 200 and 300, unplug the drive's power cable and plug it back in.
- On the Series 400, follow the procedures in sections "Normal Environment Shutdown" and "Normal System Power-Off" in Chapter 2, and then restart the system.

CONSIDERATIONS FOR SERIES 200 DUAL TAPE-DRIVE CONFIGURATIONS

This section covers differences in the tape interface that result when an optional 8-mm cartridge-tape drive is added to a Series 200 machine that is already equipped with a 9-track tape drive. Most of the information covered in previous sections of this chapter apply to dual tape drives as well as single tape drives.

Recommendations for Assigning Tape Drives

When two kinds of tape drive are available on a single machine, the system manager must decide in advance which tape drive to use for system backups.

Important Considerations When Assigning Drives

- Incremental backups (primary and secondary) must be made on the same kind of tape drive as the full backup on which they are based. Full and incremental backups made on different kinds of tape drive are incompatible and cannot be restored together. Thus, if you make a full backup on an 8-mm cartridge-tape drive, you must make all subsequent primary backups on that tape drive. If you want to switch to the 9-track tape drive, you must start by making a full backup on that drive.
- The drive number and primary use for the 8-mm cartridge-tape drive differs from that for the 9-track tape drive.
- The way the backup index is stored on an 8-mm cartridge differs from the way it is stored on a 9-track tape:
  - Backups made on an 8-mm cartridge typically require just one tape cartridge. The data is written first and then the backup index is written on the same cartridge, immediately following the end of the data.
  - Backups that are made on 9-track tape drives require a minimum of two tapes. The data is written first on as many tapes as required to hold all the data. Then the backup index is written on a separate tape. (In previous Environment releases, this additional tape was called the *blue tape*. It is now referred to as the *backup-index tape*.)
- The way a backup is restored onto the system from 8-mm cartridges differs from the way it is restored from 9-track tapes:
  - When a backup is restored from 8-mm cartridges, the tape-mount request prompts you for the first tape cartridge. The system scans through the data until it finds the backup index just past the end of the data. If the backup index is not on the first cartridge, the system scans all the way to the end of the tape, rewinds it, and then prompts you for the next tape cartridge. The system scans through the data on subsequent cartridges until it finds the backup index. After the system finds and reads the backup index, the system rewinds the current tape cartridge. If this was the first tape cartridge, it reads the data automatically. If this was not the first tape cartridge, the system prompts you for the first tape cartridge. After reading the data on the first tape cartridge, the system prompts you for any remaining tape cartridges, in order.
  - When a backup is restored from 9-track tapes, the tape-mount request prompts you for the backup-index tape first. The tape-mount request then prompts you for the remaining tapes, in order, starting with the first data tape.

Recommended Drive Assignments

- The 8-mm cartridge-tape drive is the recommended choice for backups and archives. The 8-mm drive should be assigned drive number 0, making it the default tape drive and thereby simplifying the entire backup operation. A typical backup requires only one tape cartridge, allowing automation of the entire backup operation.
- The 9-track tape drive is recommended for specialized jobs, such as transfer of data to other systems that use compatible 9-track formats. The 9-track drive should be assigned drive number 3.

Interface Changes for Dual Tape Drives

All Environment tape interfaces (packages Tape, Archive, System_Backup, and so on) accommodate both the 8-mm cartridge-tape drive and the 9-track tape drive. The impact of these dual tape drives on system operations is described in the following subsections:

- Default tape drive
- Specification of tape drive through various commands
- User-written applications
- Specification of tape-drive devices
- Specification of remote devices
- Tape-related archive Options parameter
- Tape-mount requests
- Tape-related messages in the error log
- DFS backup

Default Tape Drive

Many prompts and commands require that you specify a tape drive using its logical tape-drive number. The default value for these prompts and commands is always drive number 0. Therefore, by convention, the default tape drive is the drive that has been assigned drive number 0.

When a machine has both an 8-mm cartridge-tape drive and a 9-track tape drive, the system manager must decide which drive is to serve as the default. The default drive should be the tape drive that will be used for system backups. The default drive is then assigned drive number 0, and the remaining drive is assigned a drive number in the range 1 through 3.

Logical tape-drive numbers are assigned using the IOP ENVIRONMENT menu during the boot process. The initial assignments are usually set by Rational technical representatives, although system managers can change the assignments during subsequent system-boot processes. System managers who want to do this should contact their Rational support representative for assistance.

Specification of Tape Drive through Various Commands

Several commands in package Tape have a Drive parameter through which the user can specify the tape-drive number for the desired drive. On machines with

only one tape drive, the value 0 is the only meaningful value for this parameter; on machines with two drives, you can specify any number that has been assigned to a drive.

For Tape commands that do not have a Drive parameter (such as Tape.Read and Tape.Write), you can use the To_Operator parameter to send the operator a request for a particular tape drive. The string that you enter is displayed in the Additional Info => field of the tape-mount request. However, this message field serves merely as a suggestion to the operator; the tape drive that is actually used is determined by the operator's response to the remainder of the tape-mount request.

As described in "Specification of Remote Devices," below, several commands in package Archive have a Device parameter through which the user can specify where to read or write the archive's Data and Index files. On machines with two tape drives, you can now specify the Device parameter with a fully qualified naming expression (such as "!Machine.Devices.Tape_3" to request that the operator use a particular tape drive. However, such a naming expression merely serves to display special instructions in the Additional Info => field of the tape-mount request; the tape drive that is actually used is determined by the operator's response to the remainder of the tape-mount request.

User-Written Applications

User-written applications written against Device_Independent_Io must open specific devices for reading or writing. When such applications are to perform tape operations, they must be passed naming expressions that reference a particular drive by its logical number. For example, the name "!Machine.Devices.Tape_0" references the default tape drive; the name "!Machine.Devices.Tape_3" references tape-drive number 3. The subsequent tape-mount request displays a message indicating that the specified tape drive must be used; you cannot specify a different tape drive during the tape-mount request.

In contrast, applications written against Tape_Tools do not require names that reference specific tape drives. Instead, the drive to be used is specified during the tape-mount request.

When writing tape-related applications, remember that tape marks written on 8-mm cartridge tapes occupy much more space than tape marks written on 9-track tapes. In particular, each tape mark written on an 8-mm cartridge tape occupies 2 megabytes of space. Thus, when an application writes tape marks between files, writing a large number of small files can easily exhaust the space on a single tape.

Specification of Tape-Drive Devices

Most Series 200 machines can be upgraded to support both a 9-track tape drive and an 8-mm cartridge-tape drive. When both tape drives are installed, a user can specify a naming expression for the Device parameter in the Save, List, and Restore commands in package Archive to indicate which tape drive the operator should use.

- When a machine includes only one tape drive, the special default value "Machine.Devices.Tape_0" is used to initiate a tape-mount request; any other string is interpreted as a library name.

- When a machine includes both tape drives, a user can reference a specific tape drive by replacing the default value with a fully qualified pathname. For example:

  ```
  (!Commands.Archive.List (Device => !Machine.Devices.Tape_3);)
  ```

  Note that a fully qualified naming expression begins with an exclamation mark (!); in contrast, the default special string value omits the exclamation mark.

When a tape drive is specified with the above example, the resulting tape-mount request will contain the following message field:

```
Additional Info => Use tape drive 3
```

Note that this message field merely displays a suggestion to the operator; the tape drive that is actually used is determined by the operator's response to the remainder of the tape-mount request.

When the default string value is used, it causes a tape-mount request to be displayed with no special instructions to the operator.

Specification of Remote Devices

The Device parameter in the Save, List, and Restore commands is not limited to naming a library or tape drive on the host R1000. It can specify a remote device —that is, a library or a tape drive on another R1000 on the same network. For example, assume you are logged into an R1000 called M_1, which is the machine onto which objects are to be restored. Assume further that the tape drive you want to use is the default tape drive on an R1000 called M_2, which is connected to M_1 through the network. Under these conditions, you can use the following Device parameter to specify the default tape drive on M_2:

```
Device => "!!M_2!Machine.Devices.Tape_0"
```

The device is read on M_2, and the data is sent through the network and restored on M_1. Note that if you insert the remote R1000 name in front of the default Device value "Machine.Devices.Tape_0", you also must insert an exclamation mark (!) between the R1000 name and that value.

You also can save objects to devices on other machines. For example, assume that you are entering the Save command on M_1 to save objects that reside on that machine. Then the following Device parameter causes the Data and Index files to be saved in a directory on M_2:

```
Device => "!!M_2!Some_Directory
```

Tape-Related Archive Options Parameter

The Options parameter in various Archive commands accepts tape-related options.

For example, the Archive.Save command includes a Boolean option that, when set to True, writes a tape that can be restored on an R1000 that is still running the D1 release of the Environment. Setting this option to True is necessary only if the tape will contain loaded main programs or code views. Because Environment releases are upward-compatible, a tape made with this option also can be restored on R1000s with later Environment releases.

Tape-Mount Requests

A number of commands initiate a tape-mount request on the operator's console. Such commands include commands from packages Tape and Archive, the various backup commands, and user applications written against Device_Independent_Io and Tape_Tools. When such commands are entered on an R1000 with two tape drives, the initiated tape-mount requests differ slightly from those initiated from a single-drive machine. Certain prompts and fields appear only when machines have two tape drives.

In particular, the tape-mount request initiated from a two-drive machine displays the `On which drive? [0]` prompt, which asks the operator to specify the logical number of the tape drive on which the tape operation is to take place. Note that if a particular tape drive has been specified to a Tape or Archive command, the request for that drive will appear in the Additional Info => field; the operator may, but need not, specify the requested drive at the `On which drive? [0]` prompt. This prompt does not appear on machines with only one tape drive.

Similarly, the tape-mount verification following a two-drive tape-mount request displays the Kind of Drive field along with the label information for the mounted tape. The kind of drive is given as 8MM for 8-mm cartridge-tape drives and 9_TRACK for 9-track tape drives.

For example, assume that a Series 200 has both an 8-mm cartridge-tape drive and a 9-track tape drive, and that the 8-mm cartridge-tape drive is tape drive 0 (the default) and the 9-track tape drive is tape drive 3. The following is a tape-mount request and subsequent verification in which the default 8-mm cartridge-tape drive is specified:

```
Please Load Tape
 (Kind of Tape    => CHAINED_ANSI,
  Direction       => WRITING,
  Volume Set Name => BACKUP, 07-SEP-92 16:47:23 3)
Is Tape mounted and ready to read labels? yes
On which drive? [ 0]

Info on mounted tape is

 (Kind of Drive   => 8MM,
  Kind of Tape    => UNLABELED_TAPE,
  Writeable       => TRUE)

OK to overwrite volume? [YES]
What should the volume id handling mode be? [AUTO_GENERATE]
Volume id of tape is now: 007100
```

In contrast, the following tape-mount request and verification show that the nondefault 9-track drive is specified:

```
Please Load Tape
  (Kind of Tape    => CHAINED_ANSI,
   Direction       => WRITING,
   Volume Set Name => BACKUP, 07-SEP-92 16:47:23 3)
Is Tape mounted and ready to read labels? yes
On which drive? [ 0] 3
```

```
Info on mounted tape is

  (Kind of Drive    => 9_TRACK,
   Kind of Tape     => UNLABELED_TAPE,
   Writeable        => TRUE)

OK to overwrite volume? [YES]
What should the volume id handling mode be? [AUTO_GENERATE]
Volume id of tape is now: 007100
```

Tape-Related Messages in the Error Log

In !Machine.Error_Logs.Log@, log entries pertaining to tape operations now reflect the type of drive. When the type of drive is 9-track, the entry also reports the density. Following are examples of messages reporting each kind of tape drive and density:

```
13:34:33 --- Tape_Handling Tape_Mounted Volume_Id=TEST1 ,
             Type_Of_Drive=9Track, Density=PE_1600CPI

13:34:33 --- Tape_Handling Tape_Mounted Volume_Id=TEST2 ,
             Type_Of_Drive=9Track, Density=GCR_6250CPI

12:37:05 --- Tape_Handling Tape_Mounted Volume_Id=012601,
             Type_Of_Drive=8mm
```

DFS Backup

Backups of the Diagnostic File System (DFS) can be made on the 8-mm cartridge-tape drive. If the DFS backup is to be made on a tape drive other than the default drive 0, the nondefault drive number must be specified. For example, to specify drive number 3, append "/unit=3" to the backup command:

```
CLI> backup/unit=3
```

Omitting the drive number causes the DFS backup to be made on drive 0.

Tape Diagnostics

There are several commands you can use to verify that data stored on tapes will be recoverable. These commands are covered in the following subsections:

- Tape.Examine_Labels
- System_Report.Generate
- Verify_Backup

Tape.Examine_Labels

You can verify data on an ANSI-formatted tape by executing the Tape.Examine-_Labels command with the Volume_Labels_Only parameter set to False. A diagnostic message appears in the current output whenever the command detects an inconsistency within a data segment on an ANSI-standard labeled tape. The message attempts to explain the inconsistency.

When you execute this command with the Volume_Labels_Only parameter set to True (the default value), it examines just the label of the volume, not the rest of the labels on the tape.

System_Report.Generate

The System_Report.Generate command can be used to evaluate the quality of backup tapes. A lot of errors indicates a tape is worn or faulty (or the tape drive has a problem). Note that this command displays a record showing the quality of the tape, not the quality of the backup. It does not prove a backup can actually be restored from a tape. To determine whether a backup can actually be restored from a tape, use the Verify_Backup command (see below).

Each time a tape is written or read, tape errors are logged to a system error-log file in !Machine.Error_Logs. Among these tape errors, two types are of particular interest:

- *Read-data retry errors* are logged when a read operation fails to read a given block and has to repeat the attempt. Note that a read operation is aborted on the thirteenth consecutive retry error. Thus, twelve such errors are logged before a read operation fails.
- *Write-data retry errors* are logged whenever a write operation cannot write a given block and has to traverse to another block to repeat the attempt. Note that a write operation is aborted on the thirteenth consecutive write-data retry error. Thus, twelve such errors are actually logged before a write operation fails.

The System_Report.Generate procedure scans the appropriate files in !Machine-.Error_Logs and summarizes the logged errors on a per-operation basis. The summarized information is reported in a display such as the following, where each entry represents a single read or write operation. In this example, the tape operation on volumes 043502 and 043506 had enough errors to indicate a significant problem with the quality of those two tapes. The tape operation on volume 043509 actually failed. (See the explanation following this example.)

Request Time	Mount Wait	Processing	Volume	Density	Errors
92/08/13 08:30	00:07	00:26	043500	GCR_6250CPI	143
92/08/13 09:02	00:26	00:07	043501	GCR_6250CPI	1
92/08/13 09:35	01:05	00:39	043502	GCR_6250CPI	716
92/08/13 11:19	00:20	00:07	043503	GCR_6250CPI	5
92/08/13 11:46	00:27	00:25	043504	GCR_6250CPI	3
92/08/13 12:38	00:26	00:07	043505	GCR_6250CPI	3 / 3 / 13
92/08/13 13:11	00:10	00:19	043506	GCR_6250CPI	10 / 25 / 136
92/08/13 13:40	00:05	00:36	043507	GCR_6250CPI	1 / 2 / 168
92/08/13 14:21	00:08	00:19	043508	GCR_6250CPI	69
92/08/13 14:48	00:29	00:07	043509	GCR_6250CPI	12 / 162 / 400

- `Request Time`

  Indicates the time at which the mount request for the operation was issued.

- `Mount Wait`

  Indicates how long the operator took to answer the mount request.

- `Processing`

  Indicates how long the system took to read or write the tape.

- `Volume`

  Indicates the tape label.

- `Density`

  Indicates the density with which the tape was written (for 9-track tapes only).

- `Errors`

  Displays several types of error data. If this column contains only one number, that number represents the total number of errors logged for the operation. If

this column contains a triple number, the leftmost number is the maximum number of consecutive retry errors encountered, the middle number is the total number of retry errors in the operation, and the rightmost number is the total number of errors of any kind.

As a rule of thumb, information recorded on a 9-track tape is reliable if the tape has 5 (or fewer) consecutive retry errors, 25 (or fewer) total retry errors, and 100 (or fewer) total errors (that is, 5 / 25 / 100). If a backup tape has greater numbers of errors in these categories, it is recommended that you consult your Rational support representative.

Verify_Backup

The Verify_Backup command verifies that a backup is complete and can be restored. This procedure simulates the tape operations involved in restoring a backup, verifying that all of the required labels exist and are correct. You should use this command to verify the quality of your backup, both after making your backup and later before trying to restore the Environment from it.

- When you use this procedure to verify 9-track backup tapes, you must load the *backup-index tape* (formerly known as the *blue tape*) first and then each data tape, in sequence.
- When you use this procedure to verify 8-mm cartridge backup tapes, you must load the first cartridge, which typically contains the entire backup including all the data and the backup index. If the backup spans more than one 8-mm cartridge, you will also need to load each of the other cartridges, in sequence, according to prompts displayed on the console.

The Verify_Backup procedure generates the same messages on the operations console as an actual recovery would, confirming tape operations as well as skip and read operations. If the backup is restorable from the tested tapes, the procedure concludes with the following message:

```
Successfully completed scan of backup tape
```

If the backup is not restorable from the tested tapes, the procedure concludes with the following message:

```
*** Verify_Backup DID NOT complete
```

In most cases, verifying a backup takes two to three hours. Less time may be required for verifying backups on 8-mm cartridge tapes or on lightly loaded systems. More time may be required for verifying backups on 9-track tapes or on heavily loaded systems.

The fully qualified name of the Verify_Backup procedure is !Commands.System_Maintenance'Spec_View.Units.Verify_Backup.

16 Disk Drives

This chapter includes information on disk-drive operations for the disks contained in the R1000 Series 200, Models 10, 20, and 40, and the Series 300S. Some information on disk-drive diagnostics is included to allow operations personnel to gather diagnostic data for forwarding to your Rational support representative. (See Appendix D.) This chapter does not include maintenance information, because Rational support personnel assume responsibility for disk-drive maintenance, both scheduled and unscheduled.

8-INCH DRIVES: SERIES 200, MODELS 10, 20B, and 40B; SERIES 300S

The Series 200, Models 10, 20B, and 40B, and the Series 300S contain 8-inch disk drives.

Disk-Drive Control Panel

The disk unit's control panel is shown in Figure 16-1. To gain access to the disk unit, you will need to open the door of the disk enclosure (see Figure 16-1). The control panel contains switches and indicators with which to operate the drive. These are described in Table 16-1.

Table 16-1 8-Inch Disk-Drive Control Panel

Switch or Indicator	Function
Power On	(Indicator) The indicator is lit when the DC power supply unit is functioning and the unit is turned on. The disk drives do not power up until the main R1000 power is turned on. During startup, the R1000 sequences power to the drives beginning with the highest-numbered logical unit. When power is applied to the drive, rotation of the spindle motor is enabled. Spinup takes approximately 50 seconds, at which time the Ready LED lights. Spindown takes approximately 15 seconds after the power switch is set to the Off position.
Ready	(Indicator) The Ready indicator is on when the spindle has reached the rated speed and no fault condition exists in the drive. It also lights when the initial seek is performed or when a seek or RTZ (Return-To-Zero) operation has completed.
Check	(Indicator and Switch) The indicator lights when any fault condition has occurred. The switch can be pressed to clear a fault condition. If the problem is not cleared, contact the Rational Customer Support Response Center.

Table 16-1 8-Inch Disk-Drive Control Panel (continued)

Switch or Indicator	Function
Protect	(Indicator and Switch) This indicator lights when write operations have been inhibited. When set to the On position, this switch inhibits write operations to the disk. This switch normally should be in the Off position.

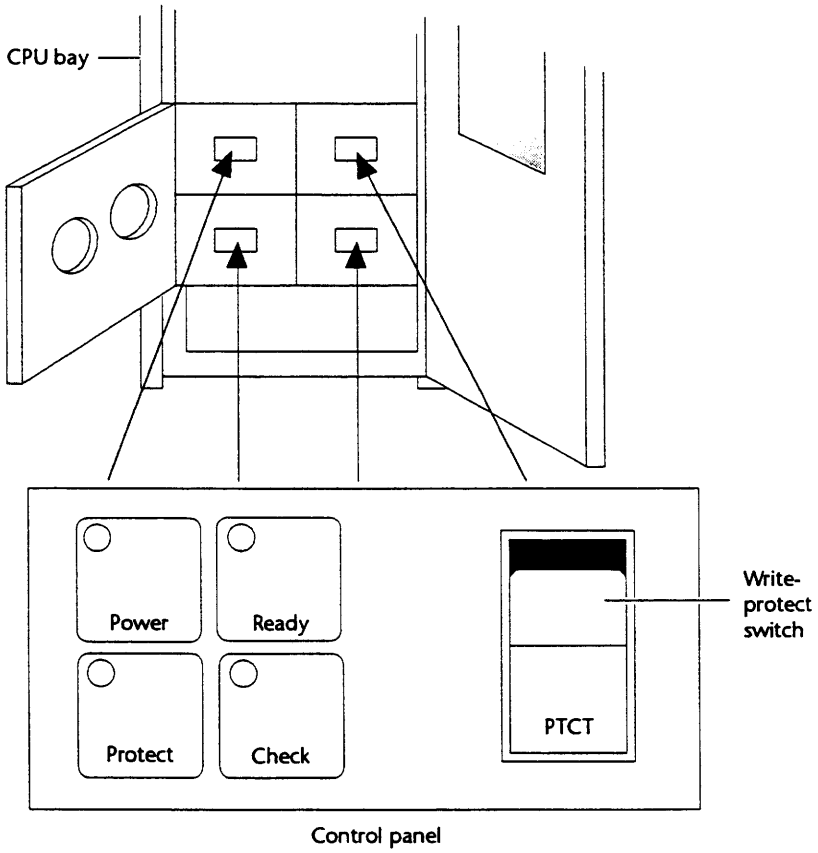

Figure 16-1 Models 10, 20B, and 40B: Disk-Drive Control Panel

Diagnostics

If the red Check indicator on the control panel is lit:

1. Check the R1000 system console to see if the system has reported any operating difficulty.
2. Attempt to clear the error by pressing the Check switch.

If the problem persists, call the Rational Customer Support Response Center.

10-INCH DRIVES: MODELS 20 and 40

The Models 20 and 40 contain 10-inch disk drives.

Disk-Drive Control Panel

The disk unit's display panel and control panel are shown in Figure 16-2. The control panel and the display panel, located next to each other, contain switches and indicators needed to operate the drive. The switches and indicators on the control panel are described in Table 16-2.

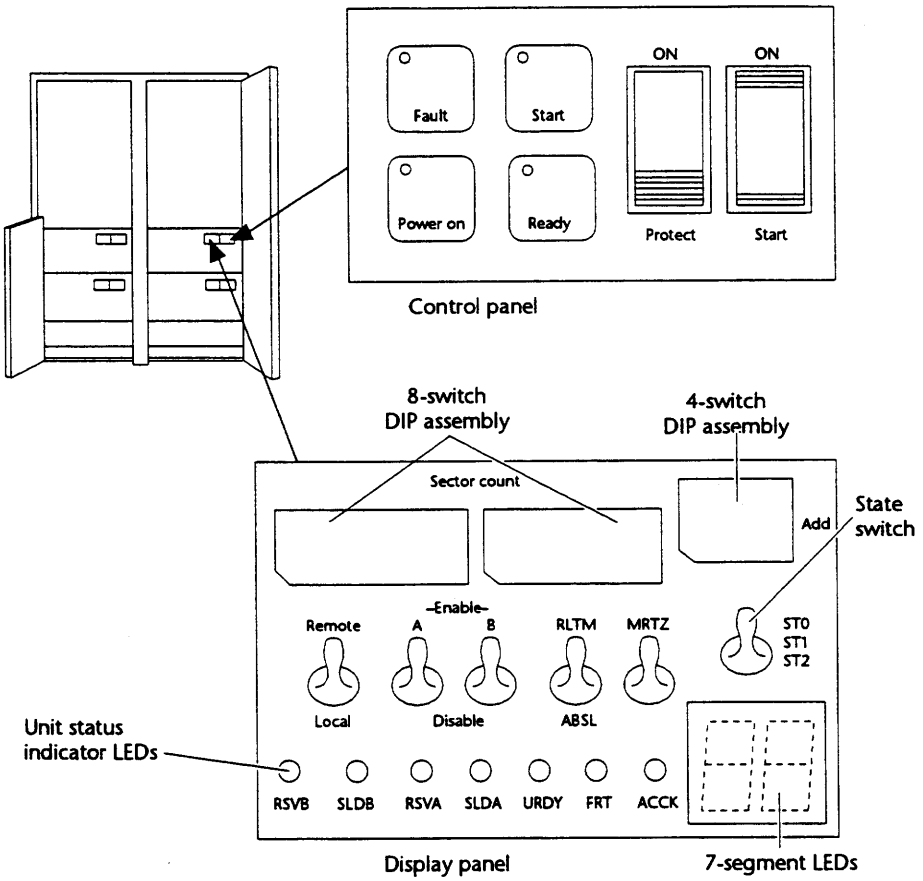

Figure 16-2 Models 20 and 40: Disk-Drive Control Panel

Table 16-2 10-Inch Disk-Drive Control Panel

Switch or Indicator	Function
Start	(Switch) When the Remote/Local switch on the display panel is set to Local, rotation of the spindle motor is enabled. Spinup takes approximately 40 seconds, at which time the Ready LED lights. Spindown takes approximately 15 seconds after the switch is set to the Off position. This switch should always be set to On and the Remote/Local switch normally should be in the Remote position.
Protect	(Switch) When set to the On position, this switch inhibits write operations on the disk. This switch normally should be set to the Off position.
Start	(Indicator) The Start indicator is on when the spindle is rotating.
Ready	(Indicator) The Ready indicator is on when the spindle has reached the rated speed and no fault condition exists in the drive.

Table 16-2 10-Inch Disk-Drive Control Panel (continued)

Switch or Indicator	Function
Fault	(Indicator and Switch) The indicator is on when a fault condition is present. The switch can be pressed to clear a fault condition. If the problem is not cleared, contact the Rational Customer Support Response Center.
Power On	(Indicator) The Power On indicator is on when the DC power supply unit is functioning. The disk drives do not power up until the main R1000 power is turned on. During startup, the R1000 sequences power to the drives beginning with the highest-numbered logical unit—that is, logical unit 2 begins spinning up first, followed by unit 1, and finally unit 0.

Diagnostics

If the red Fault indicator on the control panel is on:

1. Check the R1000 operator's console to see if the system has reported any operating difficulty.
2. Attempt to clear the error by pressing the Fault switch.

If the problem persists, call your Rational support representative. (See Appendix D.) You may be asked to check the state indicator on the display panel of the problem drive.

To check the error code:

1. Locate the display panel on the front of the drive (refer to Figure 16-2). This panel includes two seven-segment LEDs and a bank of seven individual status-indicator LEDs.
2. Locate the small three-position state switch directly above the two seven-segment LEDs. The labels ST0, ST1, and ST2 are immediately to the right of the switch.
3. Move the state switch through each of the three positions. For each position, read the status code displayed in the state indicator.
   - The normal status code is 00 when the state switch is in the ST0 position.
   - The normal status code is 07 when the state switch is in the ST1 position.
   - The normal status code is 65 when the state switch is in the ST2 position.
4. Report any display readings that are not listed in step 3 to the Rational Customer Support Response Center.

The remaining toggle switches and DIP switches on the display panel should not be changed from the configuration in which they arrived from Rational:

- For normal operation, the Remote/Local switch should be in the Remote position.
- The B switch should remain in the Disable position.
- The A switch should remain in the Enable position.
- The RLTM/ABSL switch should remain in the ABSL position.
- The MRTZ switch should not be moved.

There are seven indicator LEDs across the bottom of the display panel. During normal operation, only two of the seven LEDs should be on—namely, the LEDs labeled URDY and SLDA.

Caution: *The two 8-switch DIP assemblies and the one 4-switch DIP assembly are set to represent the sector count and the logical unit number, respectively. Changing any of these switches from their original positions may result in extended system downtime and possible loss of data.*

A Commands for the Operator's Console Command Interpreter

The operator's console command interpreter is a user interface for the R1000 system console. It allows an operator to run Environment commands such as !Commands.System_Backup.Backup, !Commands.Abbreviations.Full_Backup, and !Commands.Archive.Save from the system console, eliminating the need to maintain both an Environment console and the system console in an operations center. The other functions of the system console (such as `EEDB:` and `Kernel:`) are independent of the operator's console command interpreter.

The purpose of the operator's console command interpreter is to allow the system manager to perform common system-maintenance functions from the operator's console. It is not intended for software development. See Chapter 1 for a description of logging onto and navigating through the operator's console.

The rest of this appendix describes the most common commands you can run from the operator's console interface. Table A-2, at the end of this appendix, summarizes these commands.

COMMAND DESCRIPTIONS

The following commands, grouped by function, are frequently used from the operator's console command interpreter.

Backups

Full

Command definition:

```
Full_Backup;
```

Performs a full system backup.

```
Full_Backup("<specified time>");
```

Delays the start of a backup to the specified time. For example, if you specified `"18:00"`, the backup would begin at 6:00 P.M. local time.

Example:

```
Full_Backup;

=====>> Full_Backup; (Operator.S_2 Job 211) <<=====

A mount request has been issued and is awaiting the operator.

. . .
```

See also the System_Backup.Backup procedure in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* and the "System Backups" section of Chapter 7.

Primary

Command definition:

```
Primary_Backup;
```

Performs a primary system backup. Primary backups save all changed state since the last full backup.

```
Primary_Backup("<specified time>");
```

Performs a primary system backup at the specified time. For example, if you specified `"18:00"`, the backup would begin at 6:00 P.M. local time.

Example:

```
Primary_Backup;

=====>> Primary_Backup; (Operator.S_2 Job 211) <<=====

A mount request has been issued and is awaiting the operator.

. . .
```

See also the System_Backup.Backup procedure in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* and the "System Backups" section of Chapter 7.

Secondary

Command definition:

```
Secondary_Backup;
```

Performs a secondary system backup. Secondary backups save all changed state since the last primary backup.

```
Secondary_Backup("<specified time>");
```

Performs a secondary system backup at the specified time. For example, if you specified `"18:00"`, the backup would begin at 6:00 P.M. local time.

Example:

```
Secondary_Backup;

====>> Secondary_Backup; (Operator.S_2 Job 211) <<====

A mount request has been issued and is awaiting the operator.)

. . .
```

See also the System_Backup.Backup procedure in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* and the "System Backups" section of Chapter 7.

Information

Command definition:

```
System_Backup.History;
```

Displays information about the last ten backups, showing the date and time they were performed, the kind (full, primary, secondary), and the volume ID of the tape set.

Example:

```
System_Backup.History;

-------------------------------------------------------------------
!USERS.OPERATOR % SYSTEM_BACKUP.HISTORY          STARTED 5:18:41 PM
-------------------------------------------------------------------

Incremental Backup 301 Taken At 19-SEP-92 07:53:50
                                          Based On 15-SEP-92 08:55:12
  Blue Tape Vol Name => BACKUP, 19-SEP-92 07:47:24 3
  Blue Tape Vol Id   => 061500
  Data Tape Vol Name => BACKUP, 19-SEP-92 07:47:24 3
  Data Tape Vol Id   => 061500
  Full Backup 300 Taken At 15-SEP-92 08:55:12
  Blue Tape Vol Name => BACKUP, 15-SEP-92 08:40:59 3
  Blue Tape Vol Id   => 007102
  Data Tape Vol Name => BACKUP, 15-SEP-92 08:40:59 3
  Data Tape Vol Id   => 007102

    . . .

Full Backup 292 Taken At 18-AUG-92 08:57:04
  Blue Tape Vol Name => BACKUP, 18-AUG-92 08:53:48 3
  Blue Tape Vol Id   => 061700
  Data Tape Vol Name => BACKUP, 18-AUG-92 08:53:48 3
  Data Tape Vol Id   => 061700
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Print Queues

Every physical device that is connected to the R1000, using either a communications server or the RS232C communications port labeled "comm port," is associated with a *logical device.* Logical devices are named "Terminal_*XXX*", where *XXX* is a number in the range 224–255 (except when a physical device is connected directly to the communications port, in which case the logical device is named Terminal_16).

The commands described in this section apply to printers, although logical devices are also associated with terminals, modems, and any other physical device connected to the R1000 using the communications port or a communications server.

Disable

Command definition:

```
Queue.Disable (Device : String := "");
```

Disables the print queue for the specified device. Device is the logical device for the queue, which can be determined by using the Queue.Devices command.

Example:

```
Queue.Disable ("Terminal_16");
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Enable

Command definition:

```
Queue.Enable (Device : String := "all");
```

Enables the print queue for the specified device. Device is the logical device for the queue, which can be determined by using the Queue.Devices command.

Examples:

```
Queue.Enable ("Terminal_240");

Queue.Enable;
```

The latter form enables all queues.

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Cancel Request

Command definition:

```
Queue.Cancel (Request_Id : Positive);
```

Cancels a print request. The ID of the print request can be determined by using the Queue.Display command.

Example:

```
Queue.Cancel (252);
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Device Information

Command definition:

```
Queue.Devices (Which : String := "all");
```

Displays a print queue, showing its state and the logical device to which it is assigned. The default ("all") specifies that all registered classes of print queues be shown.

Examples:

```
Queue.Devices;
```

Displays all known queues, as follows:

```
====>> Queue.Devices; (Operator.S_2 Job 231) <<====

   Device                Protocol              Characteristics   State     Classes
============  ============================  ===============  ========  =======
TERMINAL_252  TELNET (daisy_laser ( 0, 23))  Laser_Comm       Enabled   DAISY
TERMINAL_253  TELNET (mktg_laser ( 0, 23))   Laser_Comm       Enabled   MLASER
TERMINAL_250  TELNET (apple ( 0, 23))        Laser_Comm       Enabled   APPLE
TERMINAL_251  TELNET (lp_printer ( 0, 23))                    Enabled   LP
TERMINAL_254  TELNET (mktg_qms ( 0, 23))     Laser_Comm       Enabled   QMS
DEVICES       XON_XOFF                                        Disabled  (none)
```

```
Queue.Devices("LP");

====>> Queue.Devices; (Operator.S_2 Job 211) <<====

   Device                Protocol              Characteristics   State     Classes
============  ============================  ===============  ========  =======
TERMINAL_251  TELNET (lp_printer ( 0, 23))                    Enabled   LP
```

For complete information on this command, see the Queue.Classes and Queue-.Devices commands in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Request Information

Command definition:

```
Queue.Display (Class : Class_Name := "all");
```

Displays all outstanding print-queue requests for a given printer class. The default ("all") specifies that print-queue requests for all registered printer classes be displayed.

Examples:

```
Queue.Display;

====>> Queue.Display; (Operator.S_2 Job 211) <<====

ID  Time   State   Class  User            Object
==  =====  ======  =====  ====  ============================
3   12:04  Queued  LASER  SMP   !COMMANDS.SHOW'V(2)
1   11:38  Queued  LP     SMP   !USERS.SMP.COMMANDS.RUN'V(2)
2   11:39  Queued  LP     SMP   !USERS.SMP.SHOW'V(2)
```

```
Queue.Display ("LP");

====>> Queue.Display; (Operator.S_2 Job 211) <<====

ID  Time   State   Class  User            Object
==  =====  ======  =====  ====  ============================
1   11:38  Queued   LP    SMP   !RUN'V(2)
2   11:39  Queued   LP    SMP   !OPERATOR_COMMANDS.SHOW'V(2)
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Shutdown

Schedule

Command definition:

```
Schedule_Shutdown (At_Time     : String := "23:59";
                   Reason      : String := "COPS";
                   Explanation : String := "Cause not entered");
```

Schedules a system shutdown at the given time. Any valid date/time value can be supplied. The date should be of the form yr/mo/da, followed by military time of the form hr:min:sec. Specifying the date is optional; if it is not specified, 23:59 of the current date is used. The reason for the shutdown and the explanation should also be entered. Table A-1 gives the valid reasons for a shutdown:

Table A-1 Reason Codes for the Shutdown Command

Code	Explanation
Cops	Customer operations
Release	Loading of new release
Maint	Scheduled maintenance
Crash	System crashed
Hang	System was hung
Other	Other reason (give details in comments)

Examples:

```
Schedule_Shutdown ("14:00");
```

displays output on the operator's console similar to:

```
====>> Schedule_Shutdown (OPERATOR.S_1 Job 249) <<====

Shutdown Interval is 40:00.000; Archive_Enabled = FALSE
from System: 1:19:37 PM; System will shutdown in 40:00.000
```

Since a date was not specified as part of the At_Time parameter, the system shuts down at the specified time on the *current* date.

```
Schedule_Shutdown ("92/08/20 12:00");
```

displays output on the operator's console similar to:

```
====>> Schedule_Shutdown (OPERATOR.S_1 Job 244) <<====

Shutdown Interval is 18:00.000; Archive_Enabled = FALSE
from System: 11:42:33 AM; System will shutdown in 18:00.000
```

If the date is 92/08/20, then executing the command:

```
Schedule_Shutdown ("92/08/21");
```

displays output on the operator console similar to:

```
====>> Schedule_Shutdown (OPERATOR.S_1 Job 249) <<====

Shutdown Interval is 1/00:00; Archive_Enabled = FALSE
from System: 1:16:15 PM; System will shutdown in 1/00:00
```

In this case, the system will shut down one day from the time the command was executed.

See also the Operator.Shutdown and Operator.Shutdown_Warning procedures in the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* and the "Executing the Schedule_Shutdown Command" subsection in Chapter 2.

Cancel

Command definition:

```
Operator.Cancel_Shutdown;
```

Cancels a previously scheduled shutdown.

Example:

```
Operator.Cancel_Shutdown;
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.

Information

Command definition:

```
Operator.Show_Shutdown_Settings;
```

Displays the next scheduled shutdown time.

For example, if you schedule the system to shut down at 14:00 and execute the Schedule_Shutdown command at 1:19:37 P.M., executing the Operator.Show_Shutdown_Settings command displays:

```
Shutdown Interval is 40:00.000; Archive_Enabled = FALSE
Shutdown scheduled to begin in 39:25.062
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.

Terminal-Related Commands

Force Logoff

Command definition:

```
Operator.Force_Logoff (Physical_Line : Terminal.Port);
```

Logs off the user on the specified line. All uncommitted images are saved.

Example:

```
Operator.Force_Logoff (240);
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*; see also the "User Information" subsection later in this appendix to see how to display the line into which a user is logged.

Enable Login

Command definition:

```
Operator.Enable_Terminal (Physical_Line : Terminal.Port);
```

Enables the specified line for logging in.

Example:

```
Operator.Enable_Terminal (240);
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Disable Login

Command definition:

```
Operator.Disable_Terminal (Physical_Line : Terminal.Port);
```

Disables the specified line for logging in. If a user is currently logged in on the specified the line, the line will be disabled from login after the user has logged off.

Example:

```
Operator.Disable_Terminal (240);
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Terminal Settings

Command definition:

```
Terminal.Settings (Line : Port);
```

Displays information for the specified line.

Example:

```
Terminal.Settings (16);
```

```
====>> Terminal.Settings; (Operator.S_2 Job 211) <<====

Terminal Settings for Port 16
-----------------------------
Terminal Type =                    RATIONAL
Input Baud Rate =                  BAUD_9600
Output Baud Rate =                 BAUD_9600
Parity =                           NONE
Stop_Bits =                         2
Char_Size =                        CHAR_8
Flow Control for Transmit Data = NONE
Flow Control for Receive Data =  NONE
Disconnect_On_Disconnect =         FALSE
Disconnect_On_Logoff =             FALSE
Disconnect_On_Failed_Login =       FALSE
Log_Failed_Logins =                FALSE
Login_Disabled =                   FALSE
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Miscellaneous Commands

Send Messages

Command definition:

```
Message.Send (Who : String; Message : String);

Message.Send_All (Message : String);
```

Send a message to the specified user(s).

Examples:

```
Message.Send ("SMP","Please come to the 2:00 meeting");

Message.Send_All ("PM starting, Please log off");
```

For complete information on these two commands, see the Session and Job Management (SJM) book of the *Rational Environment Reference Manual.*

Set Time

Command definition:

```
Operator.Set_System_Time (To_Be : String := ">>TIME<<");
```

Sets the system time to the specified time. The time value To_Be is of the form yr/mo/da hr:min:sec.

Example:

```
Operator.Set_System_Time ("92/09/12 15:08:34");
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Show Time

Command definition:

```
What.Time;
```

Displays the current system time.

Example:

```
What.Time;
```

```
====>> What.Time; (Operator.S_2 Job 211) <<====
August 18, 1992 at 12:13:29 PM
```

For complete information on this command, see the Session and Job Management (SJM) book of the *Rational Environment Reference Manual*.

Snapshot

Command definition:

```
Daemon.Run ("snapshot");
```

Performs a system snapshot in the amount of time set by the Daemon.Snapshot-_Warning_Message procedure. The default value for this procedure is one hour.

Example:

```
Daemon.Run ("snapshot");
```

```
====>> Daemon.Run ("Snapshot"); (Operator.S_2 Job 211) <<====
92/08/18 12:14:06 +++ Running Snapshot daemon.
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual*.

User Information

Command definition:

```
Users (Jobs_Too : Boolean := False; Verbose : Boolean := False);
```

Displays information about the currently logged-in users. The Jobs_Too parameter specifies whether all jobs are shown. The Verbose parameter specifies whether the truncated or complete display is shown.

Example:

```
Users;
```

```
====>> Users; (Operator.S_2 Job 211) <<====

USER    SESSION  LINE    TIME      I_O_COUNT
=====   =======  ====  =========  ============
BLB       S_1     240    5/20:55  3756/148595
GZC_1     S_1     241  30:44.307  62110/951027
JLS       S_1     248    1:28:56  1798/67146
SMP       S_1     247    2:57:49  62150/977550
```

Daily Message

Command definition:

```
What.Message (File : String := "Daily Message");
```

Displays the daily message file.

Example:

```
What.Message;
```

Disk Space

Command definition:

```
Operator.Disk_Space;
```

Displays information for all disk volumes in the system.

Example:

```
Operator.Disk_Space;
```

```
====>> Operator.Disk_Space; (Operator.S_2 Job 211) <<====

Volume  Capacity  Available   Used    % Free
======  ========  =========  ======  ======
1        369120     264059   105061    71
2        391680     263908   127772    67
3        391680     265421   126259    67
4        401280     246396   154884    61

Total   1553760    1039784   513976    66
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual* and the "Displaying the Space Left on Disks" subsection in Chapter 5.

Def

Command definition:

```
Def (New_Context : String := "");
```

Changes the current context to the new context. This command is embedded in the operator's console command interpreter. The operator's console command interpreter, like all jobs, has a current context. This is an object (usually a library, but it can be a file or an Ada unit) within whose context names are resolved. The initial value is normally your home library, but a login procedure may have changed your context. To determine your current context, you can use the !Commands.Library.Context procedure. The Def command changes the context of the operator's console command interpreter, so that the context in which subsequent commands are executed is changed.

If the null string is used, the current context is displayed.

Example:

```
Def ("!");
92/08/18 12:14:06 ::: [Library.Context ("!", PERSEVERE);].
92/08/18 12:14:06 +++ Current context is now !
92/08/18 12:14:06 ::: [End of Library.Context command — No errors]
```

Quit

Command definition:

```
Quit;
```

Terminates the session, logging the user off. After the user has been logged off, the operator's console command interpreter again prompts for a username.

Typ

Command definition:

```
Typ (Object : String := "");
```

Displays the image of the given object. The object name can contain wildcards, in which case multiple objects can be displayed.

Create New User Account

Command definition:

```
Operator.Create_User (User     : String := ">user name<";
                      Password : String := "");
```

Creates a user with the specified name and password.

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

Example:

```
Operator.Create_User ("gzc","secret");
```

creates a user called GZC with a password of "secret."

Delete User Account

Command definition:

```
Operator.Delete_User (User : String := ">user name<");
```

Deletes the specified user. Future logins are not allowed. The user's home world is not destroyed and can be saved, moved, or deleted later.

Example:

```
Operator.Delete_User ("gzc","secret");

====>> Operator.Delete_User (OPERATOR.S_1 Job 230) <<====
92/08/17 12:26:00 +++ gzc has been deleted login will be denied.
92/08/17 12:26:00 +++ gzc's home directory will not be deleted.
```

For complete information on this command, see the System Management Utilities (SMU) book of the *Rational Environment Reference Manual.*

COMMAND SUMMARY

Table A-2, which is continued on the next page, summarizes some of the most common operator's console commands.

Table A-2 Operator's Console Command Summary

Command	Function
Saving System Information	
Full_Backup	Performs a full system backup.
Primary_Backup	Performs a primary system backup. Primary backups save all changed state since the last full backup.
Secondary_Backup	Performs a secondary system backup. Secondary system backups save all changed state since the last primary.
System_Backup_History	Displays information about the last ten backups (date and time they were performed, volume ID, kind of backup).
Archive.Save	Performs an Archive.Save of the object(s) specified.
Archive.Restore	Performs an Archive.Restore operation.
Print Queue	
Queue.Disable	Disables the print queue for the specified device (the logical device is determined using Queue-.Devices).
Queue.Enable	Enables the print queue for the specified device (the logical device is determined using Queue-.Devices).
Queue.Cancel	Cancels a print request (print request is determined using Queue.Display.)
Queue.Devices	Displays a print queue, showing its state and the logical device to which it is assigned.
Queue.Display	Displays all outstanding requests for a given class of print queues.
Shutdown	
Schedule_Shutdown	Schedules a system shutdown to be executed at the given time. Any valid date/time value can be supplied. The date should be of the form yr/mo/da, followed by military time of the form hr:min:sec.
Operator.Cancel_Shutdown	Cancels a previously scheduled shutdown.
Operator.Show_Shutdown_Settings	Displays the next scheduled shutdown time.
Terminal-Related	
Operator.Force_Logoff	Causes the user logged into the specified line to be logged off. All uncommitted images are saved.
Operator.Enable_Terminal	Enables the specified line (terminal port) for logging in.
Operator.Disable_Terminal	Disables the specified line (terminal port) for logging in.
Terminal.Settings	Displays information for the specified line (terminal port).

Table A-2 Operator's Console Command Summary (continued)

Command	Function
Miscellaneous	
Message.Send	Sends a message to the specified user.
Message.Send_All	Sends a message to the specified users.
Operator.Set_System_Time	Sets the system time to the specified time. The time value To_Be is of the form yr/mo/da hr:min:sec.
What.Time	Displays the current system time.
Daemon.Run	Performs a system snapshot in the amount of time set by the Daemon.Snapshot_Warning_Message procedure.
Users	Displays information about the currently logged-in users. The Jobs_Too parameter specifies whether all jobs are shown.
What.Message	Displays the daily message file.
Operator.Disk_Space	Displays information for all disk volumes in the system.
Def	Changes the current context to the new context. This command is unique to the operator's console command interpreter.
Quit	Terminates the console command interpreter session.
Typ	Displays the image of the given object. The object name can contain wildcards.
Operator.Create_User	Creates a user with the specified name and password.
Operator.Delete_User	Deletes the specified user. Future logins are not allowed.

B Diagnostic Crash Procedures for Release D_12_7_3 (Delta 3.1)

This appendix describes:

- How to recover from a crash, given the enhancements of release D_12_7_3 of the Rational Environment to the analysis of R1000 crashes. In particular:
  - Much of the process has been automated so that the system captures as much information as possible and allows system managers to set the system to reboot automatically after a failure.
  - The system now automatically performs the basic crash analysis and tests that previously required user intervention. The system writes the results of these tests to files that can be sent or faxed to the Rational Response Center after the system reboots. When necessary, the system stops at a menu and offers a recommended action as the default.
  - The system now automatically notifies users after a system failure and reboot, using electronic mail and a list of users in a new reboot configuration file.
- How to configure system reboot options.

WHEN YOUR SYSTEM CRASHES

When your system crashes, you will perform one or more of the actions described in this section. At a high level, you might:

- Start or view the reboot process
- Make a crash-dump tape
- Capture the context of the crash
- Recover crash information from the Environment
- Contact the Rational Response Center
- Other operations as directed by the Rational Response Center

Starting the Reboot Process

1. Check the system console to see what is displayed:
   - If the Operator Mode keyswitch is set to Interactive, the Operator Mode menu, shown below, should be displayed:

```
OPERATOR MODE MENU - options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP CONFIGURATION
    3 => Enable manual crash debugging (EXPERTS ONLY)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot SYSTEM
```

```
Enter option [Boot SYSTEM] :
```

In this case, press [Return] to enter the default (`Boot SYSTEM`) and continue with step 2.

- If the Operator Mode keyswitch is set to Automatic, the display should look as described in step 2.
- In any other case, contact the Rational Response Center (see page 335 in Appendix D).

The system analyzes the outage or failure, dumps the information to its tombstone files, displays some informational messages, and then displays the reason for the crash in reverse video below a line of equals signs (=).

2. Verify that output similar to the following occurs:

```
Logical tape drive 0 is an 8mm cartridge tape drive.
Logical tape drive 1 is declared non-existent.
Logical tape drive 2 is declared non-existent.
Logical tape drive 3 is declared non-existent.
Booting I/O Processor with Bootstrap version 0.4

Initializing M400S I/O Processor Kernel 4_2_14
Disk  0 is ONLINE and WRITE ENABLED
IOP Kernel is initialized
Initializing diagnostic file system ... [OK]
=====================================================
Restarting system after operator console BREAK key
                          -- the above line is in reverse video
```

After this, one of the following should occur:

- The system runs additional tests and then reboots on its own. This normally occurs if Failure Reboot is enabled. See the "Configuring Reboot Options" section on page 312 for information on configuring and enabling this option.

  In this case, continue with the "Capturing the Context of a Crash" section on page 307.

- The system displays the CLI/Crash menu with its recommended action as the default action:

```
CLI/CRASH MENU - options are:
  1 => enter CLI
  2 => make a CRASHDUMP tape
  3 => display CRASH INFO
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [make a CRASHDUMP tape] :
```

  In this case, continue with step 3.

- In any other case, call the Rational Response Center.

3. Press [Return] to enter the default value. In general, the default is one of the following three values:

- `enter CLI`:

  The system typically makes this recommendation if there has been some kind of hard failure on the system. If the system makes this recommendation, call the Rational Response Center (see page 335 in Appendix D).

- `Boot STANDARD configuration:`

  This recommendation usually indicates only that Failure Reboot is disabled. Usually the confidence tests run by the system have passed and no crash dump is recommended. In this case:

  a. Press [Return] to enter the recommended default and boot the system.

  The system generates messages such as the following:

```
--- Booting the R1000 Environment ---
>>>messages overwrite each other here for a while<<<
Loading Register Files and Dispatch Rams .... [OK]
Loading Control Store ......
...
```

  See Chapter 2 for a detailed description of the standard boot process.

  b. Continue with the "Capturing the Context of a Crash" section on page 307.

- `make a CRASHDUMP tape:`

  Make a crash-dump tape as described in the next section.

Note: *If your system is not booting or stops at a place other than those described in this document, please call the Rational Response Center for instructions (see page 335 in Appendix D).*

Making a Crash-Dump Tape

The system offers this recommendation only when it has determined that a crash dump may be of benefit in analyzing the problem. Even if your site does not allow crash-dump tapes to be sent to the Rational Response Center, you should make a tape; it may be possible for a Rational representative to analyze the tape at your site.

To make a crash-dump tape:

1. Get a tape of the proper type and size:
   - If you have a 9-track tape drive, use a 2,400-foot 9-track tape.
   - If you have an 8-mm tape drive, use an 8-mm tape cartridge.
2. Make sure that the tape is write-enabled:
   - If you are using a 9-track tape, insert the write-enable ring into the hub of the tape reel.
   - If you are using an 8-mm tape, slide the write-protect tab away from the center of the cartridge.
3. Mount the tape.
4. At the CLI/Crash menu prompt:

   `Enter option [make a CRASHDUMP tape] :`

   press [Return] to select the default (`make a CRASHDUMP tape`). The system responds with the following question:

   `Should the R1000 boot automatically after the crashdump finishes [y] ?`
5. Press [Return].

   The system responds by displaying:

   `Tape unit number (0..3) [0] ?`

6. Enter the unit number of your tape drive and press [Return].
   - If you are making the crash-dump tape on a 9-track tape drive, the tape drive is probably configured as unit 0.
   - If you are making a crash-dump tape on an 8-mm tape drive, the tape drive is probably configured as unit 0 on a Series 300 or 400 system or unit 3 on a Series 200 system.
7. The system may respond by displaying:

   ```
   Tape density is currently 6250 BPI,
   do you want to change it [N] ?
   ```

   If so, verify the density setting:
   - If you are using a 9-track tape drive, verify that its density is set to 6,250 BPI.
   - If you are using an 8-mm tape drive, the density is not selectable.
8. Press [Return] to continue. The system responds by displaying:

   ```
   Volume Id, (1..6 characters) ?
   ```
9. Enter a six-character tape name (volume identification). For example: `910704`.
10. When prompted for comments with a right parenthesis, `)`, prompt, enter comments that could help someone isolate or reproduce this problem. Include the following in your comments:
    - The machine's name and cluster ID.
    - The date and time, using this format: 31 Mar 92 22:17.
    - The reason that the system crashed or started to reboot (found in the reverse-video message and in the `Restarting system after ...` message).
    - The name of the company or organization that owns/operates the R1000.
    - The location of the R1000 installation.
    - Your name.
    - The information that you have gathered so far about the state of the machine and what caused the problem (except for the trace information).
11. End your comments by entering, on a new line, a second right parenthesis next to the `)` comment prompt. For example:

    ```
    ) Mariner, 902469
    ) 31 Mar 92 22:17
    ) WCS TAGSTORE PARITY ERROR
    ) ACME Widgets, Inc.
    ) Rockville, MD
    ) J. Smith
    )
    ) There were no users on the machine.
    ) A backup was in progress.
    ) There had been a power failure two hours earlier.
    ))
    ```

    Several messages are displayed while the crash dump is made. Wait for the crash dump to complete (about 15 minutes). The system automatically unloads your tape when it is done.

After the system finishes making the tape, it either:

- Reboots the system if you asked for an automatic reboot in step 5
- Returns you to the CLI/Crash menu prompt:

  ```
  Enter Option:
  ```

  Enter the choice to boot in the standard configuration and press [Return].

12. Prepare a label for the tape with the word "Crashdump" and the following items:
    - The machine's name and cluster ID.
    - The date and time, using this format: 31 Mar 92 22:17.
    - Your site's configuration. This can be determined later by executing the Show_Default command from the `EEDB:` prompt.
    - A brief, one-line summary of why the system crashed or started to reboot.
    - The Rational log ID number, if you have received one from the Rational Response Center.
13. Dismount and store the tape as follows:
14. Remove the tape from the tape drive and put the label on the 9-track tape reel (or on the 8-mm tape cartridge).
15. Remove the write-enable ring from the 9-track tape reel (or slide the write-protect tab toward the center of the 8-mm tape cartridge), thereby write-protecting the tape.
16. Put the 9-track tape reel or (8-mm cartridge) in a safe storage container.

    At this point you have saved the volatile system state, both in your observations and on the crash-dump tape.
17. Continue with the next section, "Capturing the Context of a Crash."

Capturing the Context of a Crash

Write down the following information:

- How long had the R1000 been running since the last boot?
  Was the R1000 still booting when it crashed?
  Why was it booting?
- Was the Environment up at the time?
  Was there anything abnormal about the way the Environment was functioning?
  What were users doing? Any unusual activities?
  Were any daemons (snapshot, daily, or weekly) running?
  Was a tape drive in use?
- Have there been any hardware, firmware, or software changes to this R1000 within the past week?
- Have there been any problems with or modifications of the network in the past week?
- What was the temperature in the computer room at the time of the crash?
- Have there been any problems with the air conditioning recently?
- Are there other computers (R1000s or not) in the same building? Did any of the other computers crash or have power or environmental problems at the same time?

Continue with the next section, "Recovering Crash Information from the Environment."

Recovering Crash Information from the Environment

After a system crash (any outage or failure) and before reboot commences, the system analyzes the crash and saves the information. This information is crucial in helping the Rational Response Center to determine the cause of the crash. This section describes how this information is stored and how to recover the information for further analysis before contacting the Rational Response Center as described on page 335 in Appendix D.

The system saves the crash-analysis information in a *tombstone file.* Tombstone files are maintained automatically by the system; you cannot directly access the information in them from the Environment.

There are four tombstone files available on the system; after each crash, the crash analysis is placed in the least-recently used file, so the system can maintain information for the four latest crashes.

Note: *A tombstone file is not produced for an outage caused by pressing the* [Break] *key when the system is at the DFS CLI level. Pressing the* [Break] *key at any time that the main Environment is booting or running does produce a tombstone file.*

During the system boot, the system transfers crash analysis information from its tombstone files into one or more of the following three places:

- The file(s) specified in the !Machine.Initialization.Local. Previous_Outage_Configuration file (see the "Configuring Reboot Options" section on page 312 for more information on configuring and using this file)
- Electronic mail sent to the users specified in the !Machine.Initialization.Local-.Previous_Outage_Configuration file (see the "Configuring Reboot Options" section on page 312 for more information on configuring and using this file).
- The system error log. You should look in this log only when the above two locations do not contain the correct information. The following section describes how to do this.

Getting Crash Information from System Error Logs

To obtain crash information from system error logs:

1. From an Environment command window, execute the following command:

   ```
   Show_Error_Log (Start => 0, Count => 0);
   ```

   This opens an I/O window and displays information similar to the following:

   ```
   First = 1
   Last  = 620
   ```

2. From an Environment command window, execute the following, using the values displayed above:

   ```
   Show_Error_Log (Start => 1, Count => 620);
   ```

   This displays the complete error log for today in the form of messages like the following:

```
...
11:59:29 --- CRASH_ANALYZER INFO  Analysis for Tombstone dated
11:59:29 ... CRASH_ANALYZER INFO  11:27:52 22-JAN-1992
11:59:30 --- CRASH_ANALYZER INFO
11:59:30 --- CRASH_ANALYZER INFO CRASH REASON (based on last
11:59:30 ... CRASH_ANALYZER INFO micro-PC)
11:59:30 --- CRASH_ANALYZER INFO ------------------------------
11:59:30 --- CRASH_ANALYZER INFO    0206: cpu microcode error;
11:59:30 ... CRASH_ANALYZER INFO    refer to
11:59:30 --- CRASH_ANALYZER INFO
11:59:30 --- CRASH_ANALYZER INFO LAST CONSOLE OUTPUT
11:59:31 --- CRASH_ANALYZER INFO -------------------
11:59:31 --- CRASH_ANALYZER INFO 2KREAD_NOVRAM_DATA.VAL
   ...
11:59:38 --- CRASH_ANALYZER INFO MACHINE CRASH, I/O ADAPTOR
11:59:38 --- CRASH_ANALYZER INFO Restarting system after
11:59:38 ... CRASH_ANALYZER INFO operations console
11:59:39 --- CRASH_ANALYZER INFO
11:59:39 --- CRASH_ANALYZER INFO R1000 cpu analysis
11:59:39 --- CRASH_ANALYZER INFO ------------------
11:59:39 --- CRASH_ANALYZER INFO UCODE TRACE (last 20 entries)
11:59:39 --- CRASH_ANALYZER INFO ----------------------------
11:59:40 --- CRASH_ANALYZER INFO Microcode trace for Wcs version
11:59:40 ... CRASH_ANALYZER INFO   M207_17
11:59:40 --- CRASH_ANALYZER INFO     8A1
   ...
11:59:43 --- CRASH_ANALYZER INFO    34CF
11:59:43 --- CRASH_ANALYZER INFO    34D0
11:59:44 --- CRASH_ANALYZER INFO  C  210
11:59:44 --- CRASH_ANALYZER INFO     206
11:59:44 --- CRASH_ANALYZER INFO
11:59:44 --- CRASH_ANALYZER INFO Micro Stack contents :
11:59:45 --- CRASH_ANALYZER INFO  0 -  34D0
   ...
11:59:47 --- CRASH_ANALYZER INFO 12 -   51F
11:59:48 --- CRASH_ANALYZER INFO 13 -   51F
11:59:48 --- CRASH_ANALYZER INFO 14 -   51F
11:59:48 --- CRASH_ANALYZER INFO 15 -   51F
11:59:48 --- CRASH_ANALYZER INFO
11:59:49 --- CRASH_ANALYZER INFO UCODE STATE
11:59:49 --- CRASH_ANALYZER INFO -----------
11:59:49 --- CRASH_ANALYZER INFO Top  => 04350C04 00000A80 (15)
   ...
12:00:01 --- CRASH_ANALYZER INFO
```

3. Send to the Rational Response Center all the lines of the system error log that contain the words "Crash_Analyzer Info."

OTHER OPERATIONS

In rare cases when you contact the Rational Response Center, you may be asked to run other procedures from the CLI to further diagnose problems. Some of these procedures are documented below for reference purposes.

Running Diskx

You should run the disk exerciser (Diskx) on all disks for at least 5 minutes. Diskx is a read-and-write test, but it uses only a small test area of each disk for writes, so it will not harm user data on the disks.

To run Diskx:

1. Enter the CLI. To enter the CLI from the CLI/Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD] : 1
   ```

2. At the `CLI>` prompt, enter `x diskx`:

   ```
   CLI> x diskx
   ```

3. Press the space bar to cause the latest output to be displayed. Output similar to the following appears:

   ```
   u0  bytes=> 1246208  Soft=> 0  Hard=> 0  C=> 705  T=> 25  S=> 6
   u1  bytes=> 1203200  soft=> 0  hard=> 0  C=> 73   T=> 14  S=> 8
   u2  bytes=> 1206272  soft=> 0  hard=> 0  C=> 611  T=> 3   S=> 24
   u3  bytes=> 1243208  soft=> 0  hard=> 0  C=> 634  T=> 11  S=> 10
   ```

4. Press [Control][G] to stop Diskx.

   The system returns you to the `CLI>` prompt.

5. Write down the results and any messages where soft or hard does not equal zero (0).

6. Enter `bye` to return to the CLI/Crash menu:

   ```
   CLI> bye
   ```

Running Checkdisk

You should run Checkdisk on all disks or at least on each disk you suspect.

Note: *Checkdisk may take up to 30 minutes per disk per pass.*

To run Checkdisk:

1. Enter the CLI. To enter the CLI from the CLI/Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD] : 1
   ```

2. At the `CLI>` prompt, enter `x checkdisk`:

   ```
   CLI> x checkdisk
   ```

3. When prompted, request that only new errors be reported.
4. Run at least one pass. The system returns you to the `CLI>` prompt.
5. Write down the HDA numbers.
6. Write down any other results.
7. Enter `bye` to return to the CLI/Crash menu:

   ```
   CLI> bye
   ```

Running the FRUs

Run the field-replaceable unit (FRU) tests by following the procedure below:

1. Enter the CLI. To enter the CLI from the CLI/Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD] : 1
   ```

2. At the `CLI>` prompt, enter `x fru`:

   ```
   CLI> x fru
   ```

   The system displays the FRU Main menu:

   ```
   Rational R1000 FRU diagnostic driver
   Initializing ...
   Main menu
       1 => Display cluster information
       2 => Execute confidence test (microdiagnostic)
       3 => Execute diagnostics
       4 => Execute PM tests (not implemented yet)
       5 => Margin cluster clocks/power
       6 => Specify test options
       7 => Repair assistance (model 100 CPU power control)
       8 => Initialize processor state
       0 => Exit
   ```

3. Enter option 3 (`Execute diagnostics`).

   The system displays the Diagnostic Execution menu:

   ```
   Diagnostic execution menu
       1 => Test the foreplane
       2 => Run all tests
       3 => Run all tests applicable to a given FRU (board)
       4 => Run a specific test
       0 => Return to main menu
   ```

4. Enter option 2 (`Run all tests`).

5. When prompted for the maximum test phase, enter 2:

   ```
   Enter maximum test phase (1-3) : 2
   ```

   The system displays a long series of messages, such as those shown here:

   ```
   Running FRU P1DCOMM...
   ...
   Phase 1 passed
   ...
   Phase 2 passed
   ```

   After about 20 minutes, the system returns you to the Diagnostic Execution menu.

   If the system halts with an error, the test has failed; skip to step 7.

6. Enter option 0 (`Return to main menu`).

   The system returns you to the FRU main menu.

7. Enter option 0 (`Exit`).

   The system returns you to the `CLI>` prompt.

8. If any of the FRU tests fail, call the Rational Response Center before attempting to reboot the system (as described on page 335 in Appendix D).

   If the FRU tests passed, return to the CLI/Crash menu by entering `bye` at the `CLI>` prompt:

   ```
   CLI> bye
   ```

9. At the CLI/Crash menu, enter option 6 (`Boot STANDARD configuration`).

   The system then attempts to boot the system (see Chapter 2 for a description of the standard boot process and messages you should see).

CONFIGURING REBOOT OPTIONS

There are three aspects to configuring reboot options:

- The Autoreboot option is configured during system installation; this is always enabled unless the Rational Response Center is working with you to diagnose a specific problem and has directed you or your Rational technical representative to reinstall the system with Autoreboot disabled.
- The Failure Reboot option, which causes the system to automatically reboot after most failures, can be enabled or disabled at any time by the system manager, as described in the "Failure Reboot" section. By default, it is disabled.
- Analysis notification consists of a file that contains information about who to notify by mail after the system reboots and where to put or send information about the failure, as described in the "Analysis Notification" section.

The crash recovery process is affected by this configuration as shown in Figure B-1.

Failure Reboot

As of the D_12_7_3 release, the boot procedure allows the automation of most of the steps taken following a crash. Much of the information needed about a crash is automatically gathered and placed in tombstone files, reducing the need for crash-dump tapes.

The machine can be configured to reboot following a failure as long as no hard failure has been detected. This feature is known as *Failure Reboot.*

As a safeguard, the Failure Reboot feature allows the specification of an interval; if two crashes occur during the specified interval, the system does not automatically reboot, but rather awaits a response from the operator. This allows single, isolated crashes to be dealt with in a timely manner, but ensures that, if more than one crash occurs during a given interval, a closer investigation can take place.

The following procedures in package Dfs control the Failure Reboot feature:

- procedure Reboot_On_Failure (Enabled : Boolean := True)

  If True, allows the system to automatically reboot after a system failure.
- procedure Reboot_On_Failure_Interval (Number_Of_Days : Integer := 7)

  Sets the number of days that must pass between two crashes for the system to allow automatic reboot.

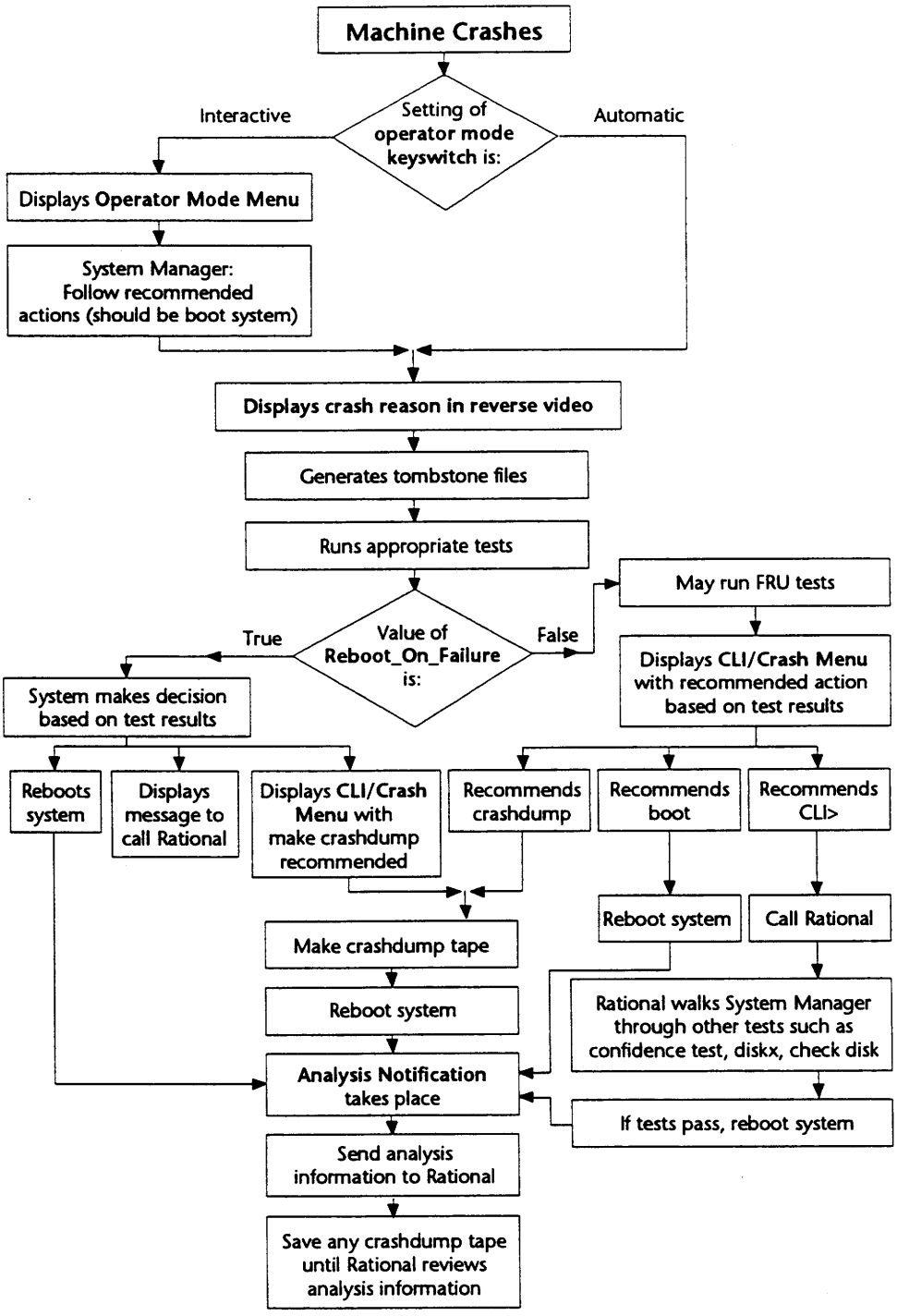

Figure B-1 Crash Recovery Process

- procedure Quiesce_Reboot_On_Failure

  Temporarily disables the Reboot_On_Failure feature for the next crash. The option is reset to enabled after the next crash or if Reboot_On_Failure is used to reenable the option.

  It is possible to quiesce only if Failure Reboot is currently enabled. If Failure Reboot is disabled, quiescing has no effect.

- procedure Reboot_On_Failure_Settings

  Displays the current settings for Failure Reboot options. The value of Failure Reboot is shown as ENABLED, DISABLED, or QUIESCED. For example:

```
The current value for the reboot interval is  1
The reboot feature is currently: ENABLED
```

As part of the Failure Reboot, the system, when applicable, automatically runs the confidence tests. If these tests detect a hard failure, the system does not reboot and prompts the operator to notify the Rational Response Center of the failure.

In the cases where no hard failure can be detected, and Reboot_On_Failure is enabled, the system reboots. At the completion of the boot, the initialization procedure Log_Previous_Outage_Start analyzes the tombstone file produced during the most recent failure.

The system writes the output of this analysis to the system error log, as well as to the file specified by the Analysis option in the Previous_Outage_Configuration file (see the following "Analysis Notification" section). The output of this analysis can then be reviewed and sent to Rational to determine the action, if any, to be taken. Possible actions could be to change the values of the Failure Reboot interval, disable Failure Reboot, use the Quiesce_Reboot_On_Failure procedure to allow the explicit creation of a crash-dump tape after the next crash, or other operations recommended by the Rational Response Center.

In the D_12_7_3 release, the Failure Reboot feature is disabled by default. The system manager or operator should enable the feature with a recommended interval of 14 days. With these settings, the system automatically reboots on the first crash, producing an analysis of the crash after the machine boots. If any subsequent crash occurs within 14 days, the system suspends rebooting and, where applicable, requests the creation of a crash-dump tape for a more complete analysis.

Analysis Notification

After a system failure occurs, the system produces an analysis of the failure and writes it to the system error log. In addition, the Previous_Outage_Configuration file allows the system manager to specify additional distribution of the analysis output. The pathname of this file is:

```
!Machine.Initialization.Local.Previous_Outage_Configuration
```

The file contains five fields:

- `Message => <<Pathname for message file>>`

  Specifies a pathname to a file that will have a brief description of the previous outage written to it. The default for this field is !Machine.Error_Logs.Crash-_Message. Only one pathname can be specified, or the field can be left blank.

- `Analysis => <<Pathname for analysis file>>`

  Specifies a pathname to a file that will have a copy of the full analysis of the previous outage written to it. The default for this field is !Machine.Error_Logs-.Crash_Analysis. Only one pathname can be specified, or the field can be left blank.

- `Boots => <<list of mail users>>`

  Specifies a list of users, separated by commas and optionally on multiple lines, who will be sent a copy of a brief description of the previous outage via R1000 mail. Mail is sent for all outages, including normal shutdowns. By default, this field is blank.

- `Crashes => <<list of mail users>>`

  Specifies a list of users, separated by commas and optionally on multiple lines, who will be sent a copy of a brief discription of the previous outage via R1000 mail. Mail is *not* sent for normal shutdowns. By default, this field is blank.

- `Full => <<list of mail users>>`

  Specifies a list of users, separated by commas and optionally on multiple lines, who will be sent a copy of the full analysis of the previous outage via R1000 mail. Mail is not sent for normal shutdowns. If the address for Rational support (support@rational.com) is specified with the proper network access, a copy of the full analysis is sent to the Rational Response Center. By default, this field is blank.

Here are sample contents for the file:

```
Message  => !Machine.Error_Logs.Crash_Message
                                    -- File with brief message

Analysis => !Machine.Error_Logs.Crash_Analysis
                                    -- File with full analysis

Boots    => operator -- Brief message to user operator on
                     -- all boots

Crashes  => user1,user2  -- Brief mail message to users listed
            user3        -- on boots following a crash

Full     => Support@Rational.Com
                -- Mail message sent with full analysis of
                -- crashes
```

C Diagnostic Crash Procedures for Release D_12_6_5 (Delta 3.0)

This appendix describes enhancements to the process of analyzing R1000 crashes. In particular:

- Much of the process has been automated so that the system will capture as much information as possible and often reboot automatically.
- A series of questions and menus have been added to the diagnostic crash procedures, which:
  - Capture system crash state
  - Analyze system hardware

WHEN YOUR SYSTEM CRASHES

Call the Rational Response Center whenever your R1000 fails. You will be guided through the appropriate parts of the following procedures—and, in some cases, through other undocumented steps as well—to reach an effective and timely diagnosis of the problem. For information about contacting the Rational Response Center, see Appendix D.

If your system crashes outside Rational's normal support hours, follow one of the three options described below. These options are listed in order, from most desirable to least desirable (in terms of gathering information needed to determine why your machine crashed and to prevent additional crashes):

- Wait and contact the Rational Response Center during normal support hours (see Appendix D).
- Gather diagnostic information about your system crash (by following the diagnostic crash procedures in this appendix) before you try to reboot the machine.
- Try to reboot the machine immediately without gathering diagnostic information. Be aware that doing so may lengthen the time it takes to isolate and resolve the problem.

USING THESE DIAGNOSTIC CRASH PROCEDURES

The instructions in this appendix are designed to help you collect as much diagnostic information as possible about the state of your machine at the time of a system crash. By following these procedures, you will help shorten the time to diagnose and, if necessary, repair your system.

These procedures are organized into sections roughly according to the order in which you may need to perform them.

No matter why the system crashed, you should always capture information about the system state, as described in the following section.

Then, if you experienced or are experiencing:

- A power failure, see "Crash due to Power Failure."
- Overheating, see "Crash due to Overheating."
- The "connecting modem" message, begin with "Display of 'Connecting Modem' Message Followed by a System Hang."
- Any other failure message, begin with "Other System Failures."
- Disk errors, begin with "Other System Failures" and continue through "If You Suspect Disk Errors."

After executing the procedures in the following sections, fax or e-mail the information you have gathered to the Rational Response Center.

Note that when the R1000 CPU stops running, the I/O processor queries all the boards to determine what caused it to stop. The message given by the R1000 when it crashes names the board that detected the specific check; therefore, the named board is not necessarily the faulty part.

SAVING INFORMATION ABOUT THE SYSTEM STATE

When your system fails, you should begin by saving information about the system state.

1. Write down the crash information displayed on the operator's console. If this information has already scrolled off the screen, go to step 2. In particular, write down the machine check message (MC), crash reason (CR), trace (TR), and any other messages (OM) displayed. A sample crash output is shown below:

```
MC=>  Sequencer has detected a machine check.
      ****************************************************************
CR=>  Booting R1000 IOP after halt of machine crash detected
CR=>  Boot reason code = 0C, from PC 0001A094
      Restarting R1000-200 August 14th, 1992 at 14:14:35

      -- you may see the Interactive menu here. If so, press [Return].

      Logical tape drive 0 is a 8MM cartridge tape drive.
      Logical tape drive 1 is declared non-existent.
      Logical tape drive 2 is declared non-existent.
      Logical tape drive 3 is a 1/2 inch 9 track tape drive at physical unit 0.
      Booting I/O Processor
      IOC Bootstrap Version 5.7

      Initializing I/O Processor Kernel 0_8_7
      Disk Controller 0, Disk  0 is ONLINE and WRITE ENABLED
      Disk Controller 0, Disk  1 is ONLINE and WRITE ENABLED
      Disk Controller 0, Disk  2 is ONLINE and WRITE ENABLED
      Disk Controller 0, Disk  3 is ONLINE and WRITE ENABLED

      IOP Kernel is initialized
      Initializing diagnostic file system ... [OK]
         FREEZE_WORLD.FIU -- Several messages like these overwrite
         MF.IOC           -- each other
```

```
==================================================
CR=>  Restarting system after R1000 Processor Halt => Processor multi-bit memory error
      -- Previous two lines indicate the crash reason and appear in reverse video.

      Running post-mortem diagnostics.
TR=>  uaddr trace:  4=2A92  3=018D  2=020E  1=0204 -- ucode M207_36
CR=>  Microcode HALT
CR=>  The processor got a(n) Halt.
OM=>  Post-mortem diagnostics do not apply to this kind of failure

      Saving state of: board JKLMFQTVI, Special Registers, Trace Rams [OK]
      R1000 Crash Save done - tombstone file R1000_DUMP1 created.
```

2. Write down the context of the crash.
   - How long had the R1000 been running since the last boot?
     Was the R1000 still booting when it crashed?
     Why was it booting?
   - Was the Environment up at the time?
     Was anything abnormal about the way the Environment was functioning?
     What were users doing? Any unusual activities?
     Were any daemons (snapshot, daily, or weekly) running?
     Was a tape drive in use?
   - Have there been any hardware, firmware, or software changes to this R1000 within the past week?
   - Have there been any problems with or modifications of the network in the past week?
   - What was the temperature in the computer room at the time of the crash?
   - Have there been any problems with the air conditioning recently?
   - Are there other computers (R1000s or not) in the same building?
     Did any of the other computers crash or have power or environmental problems at the same time?
3. Use the crash reason message (displayed in reverse video below the line of equals signs) to locate the appropriate recovery process in the "Common Crash Reasons and Responses" section.

CRASH DUE TO POWER FAILURE

Note: *A crash due to a power failure does not cause the system to dial Rational through the system's internal modem, even when the system is configured to do so for other types of crashes.*

Indication of Power Failure

In most cases, a crash due to power failure is reported by the following message in reverse video on the operator's console:

```
Restarting system after loss of AC power
```

Recovering from Power Failure

When your system crashes because of a power failure:

1. If power to the system is still off, turn the system's power keyswitch to OFF.

   Turning the power keyswitch to OFF reduces the chance of accidental damage to the R1000 when the source of power is restored.

2. *Before* trying to restart the system, try to determine the cause and extent of the power failure and to correct the problem.

   Determining the cause and extent of the power failure commonly requires the aid of your site-facilities personnel. If the power failure affected your entire site, your site-facilities personnel should contact your local power company for further information.

3. Restore power to the R1000 and turn the power keyswitch to ON.
4. After power has been restored, the system will do one of three things, depending on its configuration:
   - Boot automatically
   - Begin booting and stop at the Interactive menu:

     ```
     Options are:
         1 => Change BOOT/CRASH/MAINTENANCE options
         2 => Change IOP ENVIRONMENT configuration
         3 => Enable manual crash debugging (experts only)
         4 => Boot IOP, prompting for tape or disk
         5 => Boot system
     Enter option [Boot System]:
     ```

     If the system stops at the Interactive menu, press [Return]. The system will display the Crash menu.

   - Begin booting and stop at the Crash menu:

     ```
     Options are:
       1 => Enter CLI
       2 => Make a CRASHDUMP
       3 => Run the FRU tests
       4 => Boot DDC configuration
       5 => Boot EEDB configuration
       6 => Boot STANDARD configuration
     Enter option [Boot STANDARD] :
     ```

     If the system stops at the Crash menu, press [Return]. The system will continue booting.

5. If the system does not reboot properly, then:
   a. Examine the operator's console to see what messages are displayed.
   b. Contact the Rational Response Center (see Appendix D).

CRASH DUE TO OVERHEATING

Note: *A crash due to overheating does not cause the system to dial Rational through the system's internal modem (mounted on the PCM card in the PCM unit), even when the system is configured to do so for other types of crashes.*

Indications of Overheating

In most cases, a crash due to overheating causes all of the front-panel LEDs to be off although the power switch is on.

You can determine whether the failure is due to overheating by examining the state of the LED overtemp indicator. On Series 200 and 300 systems, the overtemp indicator is on the power control and modem (PCM) unit and can be viewed by opening the rear door of the CPU cabinet. On a Series 400 system, the indicator is on the I/O panel, visible by opening the front door of the cabinet.

Warning: *Do not reach inside the CPU bay. Electric-shock hazards are present inside the R1000 cabinets.*

If the overtemp indicator is lit, this means that sensors in the R1000 detected that the temperature inside the cabinet was too high and powered the system off.

Recovering from Overheating

1. Check the room temperature. Note that the room air temperature must not exceed 85°F (29°C) while the R1000 is operating. (For detailed environmental requirements, refer to the *Site-Planning Guide* for your system.)
2. If the room temperature is correct, the problem might be a fan failure inside the cabinet, clogged filters, or some other problem. In this case:
   a. Verify that the power keyswitch on the control panel is in the OFF position.
   b. Contact the Rational Response Center for assistance (see attached cross reference).
3. After the overheating condition has been corrected, power off and restart the system by following the instructions in steps 3 through 5 of the "Recovering from Power Failure" section, above.
4. If the system does not reboot properly:
   a. Examine the operator's console to see what messages are displayed.
   b. Contact the Rational Response Center (see Appendix D).

DISPLAY OF "CONNECTING MODEM" MESSAGE FOLLOWED BY A SYSTEM HANG

The system crashes, displays a reverse-video error message on the operator's console, and then starts to reboot. Midway through the boot process, the system pauses and dials the Rational Response Center, displaying

```
Connecting modem
```

Wait at least 3 minutes for the call to complete. If you see the message:

```
Rational Response Center has instructed this cluster to wait for
call back
```

and do not want to wait, you can proceed with booting the system as follows:

1. Press [Break] on the operator's console.
2. When the `Enter option` prompt is displayed, press [0] and [Return] and then [Y] and [Return].

   The modem connection is broken and the boot process begins.
3. Continue with the "Other System Failures" section, below.

Note: *Systems with restricted-access support contracts may not be configured to use this dial-out capability.*

OTHER SYSTEM FAILURES

When your system crashes, it may display one or two menus or continue to boot, depending on the setting of the Operator Mode key, the setting of the boot/crash/maintenance options, and the type of crash:

- If the Operator Mode key is set to Interactive, the system displays the Interactive menu:

```
Options are:
    1 => Change BOOT/CRASH/MAINTENANCE options
    2 => Change IOP ENVIRONMENT configuration
    3 => Enable manual crash debugging (experts only)
    4 => Boot IOP, prompting for tape or disk
    5 => Boot system
Enter option [Boot System]:
```

  Press [Return]. This will take you to the Crash menu.

- If either the Auto Crash Recovery or the Auto Boot option is set to No, or if the Operator Mode key is set to Interactive and you have pressed [Return], then the system displays the Crash menu:

```
Options are:
  1 => Enter CLI
  2 => Make a CRASHDUMP
  3 => Run the FRU tests
  4 => Boot DDC configuration
  5 => Boot EEDB configuration
  6 => Boot STANDARD configuration
Enter option [Boot STANDARD] :
```

  The Crash menu provides access to the Command Line Interpreter (CLI), various diagnostic tests, and rebooting options.

The crash reason is displayed above in reverse video. Use the crash reason to find the relevant subsection of "Common Crash Reasons and Responses," below.

Redisplaying the Crash Reason

If the crash reason is not visible as captured or if you can print from the console, follow the procedures given here to redisplay the crash reason and message. The crash reasons and recovery procedures for each are described in the "Common Crash Reasons and Responses" section.

1. At the Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD]: 1
   ```

   Option 1 takes you to the CLI.

2. Enter `x tombstone`:

   ```
   CLI> x tombstone
   ```

   The console displays the Display Tombstone File menu:

   ```
   Options are:
       0 => Exit
       1 => Display tombstone file 1
       2 => Display tombstone file 2
       3 => Display tombstone file 3
       4 => Display tombstone file 4
   ```

3. Enter option 1 (`Display tombstone file 1`).

   The four tombstone files act as a ring buffer with information about the most recent crash always in file 1. After entering option 1, you will see a message and menu like the one below:

```
Analysis of tombstone 1 dated - 12:44:52 24-AUG-92

Options are:
    0 => DONE
    1 => Show all
    2 => Show last console output
    3 => Show crash classifications
    4 => Show restart output
    5 => Show trace
    6 => Show Cpu State
    7 => Show queues
```

4. Enter option 4 (`Show restart output`).

   The system displays the text shown earlier in "Saving Information about the System State."

5. Make a note of the crash reason, which is indicated in reverse video under the line of equals signs. These messages are described in "Common Crash Reasons and Responses," below.

6. If your console is set up in such a way that you can easily send its output to a file or printer and e-mail or fax the output to the Rational Response Center, then:

   - Enter option 1 (`Show all`).
   - Capture the output in a file or send it to the printer.

   If your console is not set up so that you can capture its output, you can capture this information from the Environment when your system is running again; see "Recovering Crash Information from the Environment."

7. Enter option 0 (`DONE`).

   The system returns you to the Display Tombstone File menu.

8. Enter option 0 (`Exit`).

   The system returns you to the `CLI>` prompt.

9. Enter `bye`:

```
CLI> bye
```

   The system returns you to the Crash menu.

10. Continue with "Common Crash Reasons and Responses," below. If you suspect disk problems or encountered disk errors, also see "If You Suspect Disk Errors."

11. Send the information you have gathered to the Rational Response Center (see Appendix D).

COMMON CRASH REASONS AND RESPONSES

Once you have determined the crash reason (using the procedure described in "Other System Failures." above), you can continue recovering from the crash.

Following are the common crash reasons, listed by message, and the recovery actions for each.

Halt => System Error

From the Crash menu, enter option 6 (`Boot STANDARD configuration`):

```
Enter option [Boot STANDARD] : 6
```

The system attempts to boot.

If your system will not boot, contact the Rational Response Center for further instructions (see Appendix D).

Halt => I/O Processor Hardware Error

Press the white button to reboot the machine. Choose the `Boot system` or `Boot STANDARD configuration` options in any menus that you encounter.

If the system still fails to boot or crashes again with the same reason, call the Rational Response Center (see Appendix D).

Halt => I/O Processor Software Error

Press the white button to reboot the machine. Choose the `Boot system` or `Boot STANDARD configuration` options in any menus that you encounter.

If the system still fails to boot or crashes again with the same reason, call the Rational Response Center (see Appendix D).

Halt => Processor Hardware Error

See "Halt => Processor Sysbus Hardware Error," below.

Halt => Processor Multibit Memory Error

See "Halt => Processor Sysbus Hardware Error," below.

Halt => Processor Sysbus Hardware Error

Run the confidence (microdiagnostic) and field-replaceable unit (FRU) tests by following the procedure below:

1. From the Crash menu, enter option 3 (`Run the FRU tests`):

   ```
   Enter [Boot STANDARD] : 3
   ```

   The system displays the FRU Main menu:

   ```
   Rational R1000 FRU diagnostic driver
   Initializing ...

   Main menu
       1 => Display cluster information
       2 => Execute confidence test (microdiagnostic)
       3 => Execute diagnostics
       4 => Execute PM tests (not implemented yet)
       5 => Margin cluster clocks/power
       6 => Specify test options
       7 => Repair assistance (model 100 CPU power control)
       8 => Initialize processor state
       0 => Exit
   ```

2. Enter option 2 (`Execute confidence test (microdiagnostic)`).

The system displays:

```
Preparing to run the microdiagnostic (uDIAG)
The long version stress tests the DRAMs but runs 2 minutes
longer
```

3. When prompted to run the long version, enter y:

```
Do you want to run the long version [N] ? y
```

The system displays several messages and returns you to the FRU Main menu:

```
LOAD_HRAM_32_0.FIU ...
File  : DIAG.M200_UCODE
Bound : July 15, 1992  13:02:00
Delta : RM:<MICROCODE.DIAG.IOC_DELTA>UCODE.DB.1
Mom   : RM:<MICROCODE.DIAG.IOC_MOM>UCODE.DB.4
Loading Register Files and Dispatch Rams .... [OK]
Loading Control Store ............ [OK]
   PREP_WRITE_REG.TYP...
Main menu
    1 => Display cluster information
    2 => Execute confidence test (microdiagnostic)
    3 => Execute diagnostics
    4 => Execute PM tests (not implemented yet)
    5 => Margin cluster clocks/power
    6 => Specify test options
    7 => Repair assistance (model 100 CPU power control)
    8 => Initialize processor state
    0 => Exit
```

4. Enter option 3 (`Execute diagnostics`).

The system displays the Diagnostic execution menu:

```
Diagnostic execution menu
    1 => Test the foreplane
    2 => Run all tests
    3 => Run all tests applicable to a given FRU (board)
    4 => Run a specific test
    0 => Return to main menu
```

5. Enter option 2 (`Run all tests`).
6. When prompted for the maximum test phase, enter 2:

```
Enter maximum test phase (1-3) : 2
```

The system displays a long series of messages, such as those shown here:

```
Running FRU P1DCOMM ...
...
Phase 1 passed
...
Phase 2 passed
```

After about 20 minutes, the system then returns you to the Diagnostic execution menu.

If the system halts with an error, the test has failed; skip to step 9.

7. Enter option 0 (`Return to main menu`).

The system returns you to the FRU Main menu.

8. Enter option 0 (`Exit`).

The system returns you to the Crash menu.

9. If either the confidence or the FRU test failed, call the Rational Response Center before attempting to reboot the system (see Appendix D).

   If the confidence and FRU tests passed, enter option 6 (`Boot STANDARD configuration`):

   ```
   Enter [Boot STANDARD] : 6
   ```

   The system attempts to continue booting.

Halt => Processor Microcode Error

See "Halt => Processor Crash Error," below.

Halt => Processor Software Error

See "Halt => Processor Crash Error," below.

Halt => Shutdown from Environment

This message indicates that the system was deliberately shut down with the Operator.Shutdown command. If the system was shut down because of a problem, see "Halt => Processor Crash Error," below.

Halt => Processor Crash Error

1. Make a crash-dump tape. ***Note:*** *Even if your site does not allow crash-dump tapes to be sent to Rational, you should still make the tape because it might be possible for someone to analyze it at your site.*

   a. From the Crash menu, enter option 2 (`Make a CRASHDUMP`):

   ```
   Enter option [Boot STANDARD] : 2
   ```

   b. Mount a tape.

   - If you have a 9-track tape drive, mount a 2,400-foot 9-track tape.
   - If you have an 8-mm tape drive, mount an 8-mm tape cartridge.

   c. Make sure the tape is write-enabled.

   - If you are using a 9-track tape, insert the write-enable ring into the hub of the tape reel.
   - If you are using an 8-mm tape, slide the write-protect tab away from the center of the cartridge.

   d. If your tape drive has a door, close it.

   The system responds by displaying:

   ```
   Tape unit number (0..3) [0] ?
   ```

   e. Enter the unit number of your tape drive and press [Return].

   - If you are making the crash-dump tape on a 9-track tape drive, the tape drive is probably configured as unit 0.
   - If you are making a crash-dump tape on an 8-mm tape drive, the tape drive is probably configured as unit 0 on a Series 300 or 400 system or unit 3 on a Series 200 system.

   f. The system may respond by displaying:

   ```
   Tape density is currently 6250 BPI,
   do you want to change it [N] ?
   ```

   If so, verify the density setting.

- If you are using a 9-track tape drive, verify that its density is set to 6,250 BPI.
- If you are using an 8-mm tape drive, the density is not selectable.

g. Press [Return] to continue. The system responds by displaying:

```
Volume Id, (1..6 characters) ?
```

h. Enter a six-character tape name (volume identification). For example:

```
910704
```

i. When prompted for comments with a) prompt, enter comments that could help someone isolate or reproduce this problem. Include the following in your comments:

- The machine's name and cluster ID.
- The date and time, using this format: 31 Aug 92 22:17.
- The reason the system started to reboot (found in the reverse-video message and in the `Restarting system after ...` message).
- The name of the company or organization that owns/operates the R1000.
- The location of the R1000 installation.
- Your name.
- The information about the state of the machine and what caused the problem that you have gathered so far (except for the trace information).

j. End your comments by entering, on a new line, a second right parenthesis next to the) comment prompt.

For example:

```
) Mariner, 902469
) 31 Aug 92 22:17
) WCS TAGSTORE PARITY ERROR
) ACME Widgets, Inc.
) Rockville, MD
) J. Smith
)
) There were no users on the machine.
) A backup was in progress.
) There had been a power failure two hours earlier.
))
```

Several messages are displayed while the crash dump is made. Wait for the crash dump to complete (about 15 minutes). The system automatically unloads your tape when it is done.

After the system finishes making the tape, it returns you to the Crash menu.

k. Prepare a label for the tape with the words "Crash Dump" and the following items:

- The machine's name and cluster ID.
- The date and time, using this format: 31 Aug 92 22:17.
- Your site's configuration. This can be determined later by executing the Show_Default command from the `EEDB:` prompt.
- A one-line summary of why the system started to reboot.
- The Log ID, if you have one.

l. Dismount and store the tape as follows:

i. Remove the tape from the tape drive and put the label on the 9-track tape reel (or on the 8-mm tape cartridge).

ii. Remove the write-enable ring from the 9-track tape reel (or slide the write-protect tab toward the center of the 8-mm tape cartridge), thereby write-protecting the tape.

iii. Put the 9-track tape reel or (8-mm cartridge) in a safe storage container.

At this point you have saved the volatile system state, both in your observations and on the crash-dump tape.

2. Enter option 6 (`Boot STANDARD configuration`):

```
Enter option [Boot STANDARD] : 6
```

The system attempts to continue booting.

3. Contact the Rational Response Center for instructions on having your crash-dump tape analyzed (see Appendix D).

Machine Check (Parity Error)

This message indicates that there was a problem with the CPU hardware. The confidence test may run automatically or the Rational Response Center may be called using the modem. If the test passes, the system may automatically reboot or display the Crash menu, in which case you should enter `6` (`Boot STANDARD configuration`) and press [Return]. If the confidence tests fail, see "Halt => Processor Sysbus Hardware Error."

Any Other Reason

Call the Rational Response Center (see Appendix D).

If you are unable to reach someone, then:

1. From the Crash menu, enter option 2 (`Make a CRASHDUMP`):

```
Enter option [Boot STANDARD] : 2
```

The system requests that you load a tape, enter some information, and then returns to the Crash menu (for more details, see "Halt => Processor Crash Error," above).

2. Run the confidence (microdiagnostic) and field-replaceable unit (FRU) tests by following the instructions in "Halt => Processor Sysbus Hardware Error," above.
3. If the confidence and FRU tests fail, call the Rational Response Center (see Appendix D).

   Otherwise, attempt to reboot the system.
4. Fax or e-mail the information you have gathered to the Rational Response Center (see Appendix D).

IF YOU SUSPECT DISK ERRORS

If there were any disk errors, or if you suspect disk problems, run the disk exerciser (Diskx) and Checkdisk from the CLI.

Running Diskx

You should run the disk exerciser (Diskx) on all disks for at least 5 minutes. Diskx is a read-and-write test, but it uses only a small test area of each disk for writes, so it will not harm user data on the disks.

To run Diskx:

1. Enter the CLI. To do this from the Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD] : 1
   ```

2. At the `CLI>` prompt, enter `x diskx`:

   ```
   CLI> x diskx
   ```

3. Press the space bar to cause the latest output to be displayed. Output similar to the following will appear:

   ```
   u0  bytes=> 1246208  soft=> 0  hard=> 0  C=> 705  T=> 25  S=> 6
   u1  bytes=> 1203200  soft=> 0  hard=> 0  C=> 73   T=> 14  S=> 8
   u2  bytes=> 1206272  soft=> 0  hard=> 0  C=> 611  T=> 3   S=> 24
   u3  bytes=> 1243208  soft=> 0  hard=> 0  C=> 634  T=> 11  S=> 10
   ```

4. Press [Control][G] to stop Diskx.

   The system returns you to the `CLI>` prompt.

5. Write down the results and any messages where soft or hard does not equal 0.

6. Enter `bye` to return to the Crash menu:

   ```
   CLI> bye
   ```

Running Checkdisk

You should run Checkdisk on all disks or at least on each disk you suspect.

Note: *Checkdisk may take up to 30 minutes per disk per pass.*

To run Checkdisk:

1. Enter the CLI. To do this from the Crash menu, enter option 1 (`Enter CLI`):

   ```
   Enter option [Boot STANDARD] : 1
   ```

2. At the `CLI>` prompt, enter `x checkdisk`:

   ```
   CLI> x checkdisk
   ```

3. When prompted, request that only new errors be reported.

4. Run at least one pass. The system returns you to the `CLI>` prompt.

5. Write down the HDA numbers and any other results.

6. Enter `bye` to return to the Crash menu:

   ```
   CLI> bye
   ```

RECOVERING CRASH INFORMATION FROM THE ENVIRONMENT

If your console is *not* set up in such a way that you can capture its output, you can obtain the crash information displayed in "Other System Failures" from the system error logs when the system boots.

To obtain the crash information from system error logs:

1. From an Environment command window, execute the following command:

   ```
   Operator.Internal_System_Diagnosis
   ```

   The system opens an I/O window containing an `EEDB:` prompt.

2. Enter `kernel` and press [Promote]:

   ```
   EEDB: kernel    [Promote]
   ```

   The system displays a `Kernel:` prompt.

3. Enter `show_error_log` and press [Promote]:

   `Kernel: show_error_log` [Promote]

   The system displays a range such as:

   `first entry => 1; last entry => 440`

4. Enter the first entry of `1` and press [Promote]:

   `FIRST [-10]: 1` [Promote]

5. Enter the number of the last entry (here `440`) and press [Promote]:

   `LAST [-9]: 440` [Promote]

   The system displays a message such as the following and returns you to the `Kernel:` prompt:

```
11:59:29 --- CRASH_ANALYZER INFO  Analysis for Tombstone dated
11:59:29 ... CRASH_ANALYZER INFO  11:27:52 22-AUG-1992
11:59:30 --- CRASH_ANALYZER INFO
11:59:30 --- CRASH_ANALYZER INFO CRASH REASON (based on last micro-PC)
11:59:30 --- CRASH_ANALYZER INFO -------------------------------
11:59:30 --- CRASH_ANALYZER INFO    0206: cpu microcode error; refer to
11:59:30 --- CRASH_ANALYZER INFO
11:59:30 --- CRASH_ANALYZER INFO LAST CONSOLE OUTPUT
11:59:31 --- CRASH_ANALYZER INFO -------------------
11:59:31 --- CRASH_ANALYZER INFO 2KREAD_NOVRAM_DATA.VAL
   ...
11:59:38 --- CRASH_ANALYZER INFO MACHINE CRASH, I/O ADAPTOR
11:59:38 --- CRASH_ANALYZER INFO Restarting system after operations console
11:59:39 --- CRASH_ANALYZER INFO
11:59:39 --- CRASH_ANALYZER INFO R1000 cpu analysis
11:59:39 --- CRASH_ANALYZER INFO ------------------
11:59:39 --- CRASH_ANALYZER INFO UCODE TRACE (last 20 entries)
11:59:39 --- CRASH_ANALYZER INFO ----------------------------
11:59:40 --- CRASH_ANALYZER INFO Microcode trace for Wcs version M207_17
11:59:40 --- CRASH_ANALYZER INFO     8A1
   ...
11:59:43 --- CRASH_ANALYZER INFO    34CF
11:59:43 --- CRASH_ANALYZER INFO    34D0
11:59:44 --- CRASH_ANALYZER INFO  C  210
11:59:44 --- CRASH_ANALYZER INFO     206
11:59:44 --- CRASH_ANALYZER INFO
11:59:44 --- CRASH_ANALYZER INFO Micro Stack contents :
11:59:45 --- CRASH_ANALYZER INFO  0 -  34D0
   ...
11:59:47 --- CRASH_ANALYZER INFO 12 -   51F
11:59:48 --- CRASH_ANALYZER INFO 13 -   51F
11:59:48 --- CRASH_ANALYZER INFO 14 -   51F
11:59:48 --- CRASH_ANALYZER INFO 15 -   51F
11:59:48 --- CRASH_ANALYZER INFO
11:59:49 --- CRASH_ANALYZER INFO UCODE STATE
11:59:49 --- CRASH_ANALYZER INFO -----------
11:59:49 --- CRASH_ANALYZER INFO Top  => 04350C04 00000A80 (15)
   ...
12:00:01 --- CRASH_ANALYZER INFO
Kernel:
```

6. Enter `quit` and press [Promote]:

   `Kernel: quit` [Promote]

   The system returns you to the `EEDB:` prompt.

7. Enter `quit` and press [Promote]:

   ```
   EEDB: quit    [Promote]
   ```

   The system returns you to the Environment level.

8. Send to the Rational Response Center all the lines of the system error log that have the words "Crash_Analyzer Info" in them.

D

Rational Customer-Support Services

Rational offers a variety of customer-support services. These may include: product training, telephone-based support from local or home-office Rational Response Centers, custom tool development, and a range of consulting services.

The typical support model includes on-site training of users, toolsmiths, and system managers. One of these people—usually the system manager—is designated as the on-site point of contact for questions and problems. When this on-site contact person is unable to resolve a problem, he or she can refer it to your Rational support representative, who is either a Rational technical representative or a Rational Response Center employee, as designated by your account team. Over time, the on-site contact person develops the experience required to determine how best to use the available support services.

Your Rational account team will help you identify the correct combination of support services for your particular site. The account team will also provide you with information about how and when to contact technical support representatives and response centers according to your needs.

This appendix explains how and when to submit customer-support requests and how to communicate other support-related information to Rational.

CUSTOMER-SUPPORT REQUEST LOGS

A support request submitted to a Rational Response Center is called a *customer-support request*; each request is given a log ID. The two primary methods for submitting a request are by telephone or on tape. Other appropriate methods may be established by the account team. The urgency of the request, which is defined by how quickly a response is required, determines whether you should use the telephone or tape method. There are four different categories of problems (called *priorities*) corresponding to different levels of response, as follows:

- Critical

  A critical problem is one that is very costly and/or risks that your site will not be successful with its mission; this means that progress has halted. You should expect technical support personnel to respond to critical issues within 1 CPM[1] hour.

- High

  A high-priority problem impedes progress, but it still allows people at your site to do work. You should expect technical support personnel to respond to a high-priority issue within 4 CPM hours.

[1] Contract Period of Maintenance (CPM) hours are from 8:00 A.M. to 6:00 P.M. customer local time or as otherwise noted in the support contract.

- Medium

  A medium-priority problem has some impact at your site but does not impede progress. You should expect technical support personnel to respond to such an issue within 12 CPM hours.

- Low

  Low-priority problems do not immediately affect your site. You should expect technical support personnel to respond to these issues as time permits.

Urgent requests should be transmitted directly to your designated response center by telephone.

Nonurgent requests (such as those requesting product enhancements) should be submitted online by local users with the SIMS Request program that is available on all Rational R1000s. The system manager should review these requests and decide which ones should be forwarded to Rational on magnetic tape. (See the "Making a SIMS Log Tape" subsection in Chapter 7 for more information about this process.)

Note: *All user-initiated issues handled by a system manager and transmitted to Rational using a SIMS log tape should be low-priority support requests, because it can take a comparatively long time for support personnel to respond to such requests. More urgent problems submitted with the Request program can be handled individually by the system manager using the process identified by the account team.*

Rational will communicate the status of requests to the site system manager using one of the agreed-upon means of communication.

SUPPLYING SUPPORT INFORMATION

Before you submit a request to your designated response center, you should gather detailed information that can help with the timely resolution of the request. This information will also make it possible for support personnel to track and store your request. Please be prepared to supply:

- The cluster ID for your machine

  If you do not know this identification number, look for it on the bottom-right corner of the CPU bay door, or execute the Environment Show_Machine_Id command.

- The product or part of the Environment in question

  Be prepared to tell support personnel the name of the product or program associated with the customer-support request—for example, CMVC, package Archive, Mail, Networking, and so forth. If it is available (for example, in a What.Jobs display), please have ready the product release number also.

- The Environment release number

  If you do not know this, execute the What.Version command from a command window.

- The context of the problem

  Record complete, concise details of what happened and what other programs were running at the same time. You should note any error messages that appear on the terminal or console screen.

If you call about a previously reported problem, be prepared to provide the assigned log ID for more efficient service.

SUBMITTING SUPPORT REQUESTS

Depending on your location and support contract, there may be any number of mechanisms for requesting and receiving support services. Following is a support contact template that you can fill out with the help of your account representative. You should refer to this information whenever you place a request for support services.

1. Local Support Representative
   - Name: _____
   - Address: _____
   - Telephone: _____
   - Fax: _____
   - Internet: _____
   - Other: _____
2. Local Rational Office
   - Address: _____
   - Contact: _____
   - Telephone: _____
   - Fax: _____
   - Internet: _____
   - Other: _____
3. Local Rational Response Center
   - Address: _____
   - Contact: _____
   - Telephone: _____
   - Fax: _____
   - Internet: _____
   - Other: _____
4. Rational Home-Office Response Center in Santa Clara, California
   - TCP/IP Internet: support@rational.com
   - Address: RATIONAL
     Attention: Response Center
     3320 Scott Blvd., Santa Clara, CA 95054-3197
   - Fax: Addressed to the "Response Center":
     +1-408-496-3636 or:
     +1-408-496-3637 (use this second number only if the 3636 line is down, not just busy)

     Standard CCITT Group III telephone facsimile
     9600 bps, automatic fall-back to 7200/4800/2400 bps
     Unattended, 24-hour answering

- Telephone: 1-800-533-5444 or:
  1-800-433-5444 (from anywhere in the USA or Canada)
  +1-408-496-3610 (direct dial)

Send crash-dump tapes and SIMS log tapes to the Home-Office Response Center address listed above. Note that you can set Analysis Notification to automatically send the crash analysis information or the context information you gathered above to the Rational Response Center via electronic mail. This feature is described in Appendix B on page 312.

Note: *Whenever you submit a support request using fax, mail, Internet, or telex and do not receive a reply or log ID, please telephone to verify that Rational has received your support request.*

RESOLUTION OF CUSTOMER-SUPPORT REQUESTS

Rational Response Center personnel track all customer-support requests that have been assigned a log ID. The priority of a request may change according to how critical the problem becomes and how long the request has been open.

The Rational Response Centers have many ways to resolve problems reported to them by a customer system manager or local support representative. Typical techniques for problem resolution include:

- Workarounds

  Response Center personnel may describe a workaround for a problem. For example, a workaround may require avoiding some particular sequence of keystrokes that results in a terminated session.

- Online remote diagnostics

  Remote diagnostics and problem isolation are available to some customers who do not have restricted access and who have configured their "External Modem" port and PCM board modem to allow Response Center personnel to log in remotely. In these circumstances, Response Center personnel can examine the system state online, run remote diagnostics, or execute a remote debugging session.

- Analysis of crash-dump and SIMS tapes

  Crash-dump tapes sent to the Home-Office Response Center are analyzed to determine why a system crashed. Analysis of these tapes can be especially valuable for sites that cannot allow external connections to the R1000. After the crash-dump tape is analyzed, Response Center personnel will inform the customer system manager and the account team of the results and discuss possible fixes or workarounds. (Note that crash-dump tapes can also be analyzed on site in those cases where tapes cannot be sent off site.)

  SIMS tapes compiled from the SIMS log by the system manager can provide valuable feedback from end users of the system. Problems reported in this fashion are recorded and tracked so that a suitable fix can be included in a subsequent software release.

- New releases

  Some problems can be fixed only in a new release of a software product. Rational support personnel will inform customers about such problems and help identify temporary workarounds, where possible.

- Hardware replacement

  Some critical problems, such as failure of a disk or circuit board, can be solved only by replacing the failed unit. A site system manager, the customer's account team, and the nearest response center will cooperate to replace faulty hardware as quickly as possible.

E

Machine Initialization

This appendix describes the mechanisms for initializing an R1000. This description of the machine-initialization software applies to D_12_5_0 and all subsequent releases. The software makes it easier to:

- Install layered products
- Distinguish Rational-specific, site-specific, and machine-specific customizations
- Configure a network of printers for large sites

OVERVIEW

Each time an R1000 is booted, software is executed that initializes layered products, sets various system parameters (for example, disk-collection thresholds and snapshot intervals), starts servers, enables terminals, and so on.

In Environment releases before to D_12_5_0, the boot process automatically executed !Machine.Initialize, which in turn executed a family of procedures (with names of the form !Machine.Initialize_@).

In D_12_5_0 and subsequent releases, !Machine.Initialize is no longer used. Instead, all system-initialization software resides in the !Machine.Initialization world, which is structured as shown:

```
!Machine.Initialization : Library (World);
  Local     : Library (World);
  Rational  : Library (World);
  Site      : Library (World);
  Start     : Ada (Load_Proc);
```

The D_12_5_0 (or later) boot process automatically executes the Start procedure, which in turn executes all the procedures that reside in (or are referenced in) the Local, Rational, and Site worlds:

- The Rational world contains or references software that initializes Rational products and provides standard settings for many system parameters. Users should not modify the objects in this world.
- The Site and Local worlds provide a place for system managers to put objects that customize the initialization process. Such objects can be used to override various standard system parameter settings, to initialize customer-written applications, and to specify terminal and printer configurations:
  - The Site world is intended for customer-written objects that are common to two or more machines at a given site.
  - The Local world is intended for customer-written objects that apply only to the current R1000.

The following subsections give more detail about these worlds.

!Machine.Initialization.Rational World

The Rational world contains Rational-supplied objects that perform basic initialization services for the current R1000. These objects include:

- Loaded main procedures that are executed by !Machine.Initialization.Start whenever the system is booted.
- "_Start" files that reference procedures located elsewhere in the Environment. These are text files whose names end with _Start; the procedures they reference are executed by !Machine.Initialization.Start.

On a typical system, this world contains objects such as the following:

```
!Machine.Initialization.Rational : Library (World);
  Clean_Machine_Temporary    : C Load_Proc;
  Cross_Compilers            : C Load_Proc;
  Design_Facilities          : C Load_Proc;
  Dtia                       : C Load_Proc;
  Finish_Install             : C Load_Proc;
  Log_Previous_Outage_Start  :   Text;
  Mail_Start                 :   Text;
  Network                    : C Load_Proc;
  Parameters                 : C Load_Proc;
  Printers                   : C Load_Proc;
  Servers                    : C Load_Proc;
  Teamwork_Interface         : C Load_Proc;
  Terminals                  : C Load_Proc;
```

The procedures that are supplied or referenced in this world:

- Perform cleanup and compaction
- Initialize the installed Rational products, such as Cross-Development Facility, Design Facility, Networking, and so on
- Initialize servers, including the archive and FTP servers
- Set standard values for various system parameters, such as the medium-term scheduler, snapshot intervals and warnings, and disk-collection thresholds
- Initialize terminals and printers according to user-specified requirements (given in files in the Site and Local worlds)

For more specific information, you can browse the comments in each object in this world.

!Machine.Initialization.[Site,Local] Worlds

The Site and Local worlds are where system managers can create objects to control sitewide or machine-specific initialization and configuration. These objects may include:

- Ada procedures that supplement or override the basic initialization services performed by objects in the Rational world. All procedures in the Site and Local worlds are executed by !Machine.Initialization.Start each time the system is booted.
- "_Start" files that reference procedures located elsewhere in the Environment. These are text files whose names end with _Start; the procedures they refer-

ence are executed by !Machine.Initialization.Start. Using a _Start file is equivalent to calling Program.Run or Program.Run_Job to execute the referenced procedure. (See the "Using _Start Files to Reference Initialization Procedures" section.)

- *Configuration files* that tell the Environment how to enable and configure ports for login and ports for printing. These are text files that are read by two of the procedures executed by !Machine.Initialization.Start. A default file for enabling login ports is created in the Local world during installation. (See the "Enabling and Configuring Login Ports" and "Configuring Printers" sections.)
- Text files that tell the Environment how to initialize layered products such as the Cross-Development Facility or the Rational Design Facility. (See the comments in the specifications of the Cross_Compilers and Design_Facilities procedures in the !Machine.Initialization.Rational world.)

At most sites, system managers will use the Site and/or Local worlds as a place for procedures (or _Start files) that:

- Set password policy
- Set login limits
- Start server programs for site-specific networking, databases, and applications (for example, a login monitor or network security server)

At some sites, system managers may need to use these worlds for procedures that:

- Provide nonstandard daemon settings—for example:
  - The time at which daily and weekly clients run. (The standard times are 3:00 A.M. for daily clients and 2:30 A.M. for weekly clients.)
  - How often snapshots are taken. (The standard snapshot interval is every 30 minutes.)
  - Whether (and when) to send a snapshot warning, snapshot start messages, and snapshot finish messages. (The standard is to send a warning 20 seconds before the next snapshot and to notify users only when the snapshot has finished.)
  - The interval for daemon warnings. (The standard is to send a warning 2 minutes before the daily clients begin.)
  - Whether to send disk-collection threshold warnings. (The standard is to warn users when collection thresholds have been passed.)
  - What kinds of system log messages appear on the operator's console. (The standard is to route only warning, problem, and fatal messages to the operator's console.)
  - Whether clients should perform access-list compaction. (The standard is for all relevant clients to perform access-list compaction.)
- Provide customized disk-collection thresholds (see also the "Customizing Disk-Collection Thresholds" section). Note, however, that values set by the procedure in the Rational world are calculated based on your disk configuration and should be sufficient; see your Rational technical representative if you want to use different thresholds.
- Provide nonstandard medium-term scheduler settings. (Use the Scheduler.Display command to see the standard settings.) Note, however, that values set by the procedure in the Rational world should be sufficient; see your Rational technical representative if you want to use different settings.

SETTING UP THE SITE AND LOCAL WORLDS

If you have a new R1000, you can use the following guidelines to set up your Site and Local worlds:

1. Decide whether you need to provide any system customizations such as those listed in the "!Machine.Initialization.[Site,Local] Worlds" section. Create the appropriate procedures and/or _Start files, using the hints given in the "Hints for Implementing System Customizations" section.
2. Inspect the default Terminal_Configuration file that was created in the Local world during installation. This file enables ports 235 through 249 for login. Edit this file if you want to enable a different set of ports and/or specify further connection or communication information; see the "Enabling and Configuring Login Ports" section.
3. Decide whether to implement a printer-configuration mechanism to enable users to use !Commands.Abbreviations.Print and to facilitate the use of networked printers. Create the appropriate files; see the "Configuring Printers" section.
4. If you have layered products such as the CDF or RDF, inspect the comments in the specifications of the Cross_Compilers and Design_Facilities procedures in the !Machine.Initialization.Rational world. Create any files required for initializing these products (such as a text file that registers an RDF customization).

If you are upgrading from a release prior to D_12_5_0, the Local world will already contain several objects that contain customizations:

- A Terminal_Configuration file is created automatically in the Local world, enabling the same set of ports that were enabled at the time of installation. This file also preserves any nondefault communication characteristics that were in effect for RS232 ports.
- The !Machine.Initialize_Site procedure is automatically moved into the Local world, as described in the "Installation Procedure."

Note: *After your R1000 has been upgraded to D_12_5_0 or a more recent release, you can still use the above guidelines to set up your Site and Local worlds, but first inspect the Initialize_Site procedure that has been copied into the Local world during installation. Edit this procedure to preserve any system customizations that are still appropriate. You may want to split this procedure into separate procedures and move appropriate procedures into the Site world. See the "Installation Procedure" for specific recommendations for handling this procedure. See also the "Hints for Implementing System Customizations" section for information about initialization procedures and _Start files.*

HINTS FOR IMPLEMENTING SYSTEM CUSTOMIZATIONS

This section provides information about:

- Writing customized initialization procedures in the Site and Local worlds; see the "Writing Customized Initialization Procedures" section.
- Writing _Start files that refer to procedures residing in other Environment libraries; see the "Using _Start Files to Reference Initialization Procedures" section.
- Controlling the order in which the customized initialization procedures and the _Start files are processed by the Start procedure; see the "Controlling the Order of Execution" section.

- Customizing disk-collection thresholds; see the "Customizing Disk-Collection Thresholds" section.

Writing Customized Initialization Procedures

You can write procedures in the Site or Local worlds to implement system customizations such as password policies, system daemon settings, and so on. All procedures that appear in these worlds are executed by the Start procedure each time the R1000 boots.

Customized initialization procedures can contain calls to procedures in various standard Environment packages. Some useful packages include:

- Package Daemon, which contains procedures that schedule clients and warnings
- Package Operator, which contains procedures that set password policy and login limits
- Package Scheduler, which contains procedures that control the medium-term scheduler
- Package Program, which contains procedures that execute other programs

Note that:

- You do not have to provide initialization procedures for configuring login ports and printer ports; it is recommended that you use configuration files instead (see the "Enabling and Configuring Login Ports" and "Configuring Printers" sections).
- You do not have to write an initialization procedure "from scratch" for customizing disk-collection thresholds; if you must customize these thresholds, it is recommended that you edit the sample initialization procedure provided in the Local world (see the "Customizing Disk-Collection Thresholds" section).

When writing customized initialization procedures:

- You can create separate procedures or put all calls in a single procedure. Separate procedures take longer to execute but make it easier to see what operations are being performed.
- You must not duplicate procedure names across the Rational, Site, and Local worlds.
- You can specify the relative execution order of procedures using annotations (see the "Controlling the Order of Execution" section).

Using _Start Files to Reference Initialization Procedures

As an alternative to writing procedures directly in the Site and Local worlds, you can create _Start files in one or both worlds to reference customized initialization procedures that reside elsewhere in the Environment. _Start files are processed by the Start procedure each time the R1000 boots, and the procedures they reference are executed.

When writing _Start files:

- You must choose filenames that use the _Start suffix.
- You must not duplicate filenames across the Rational, Site, and Local worlds.
- You must use annotations to reference the procedure to be executed, as illustrated below.

- You may use annotations to control the order in which the Start procedure executes the referenced procedures (see the "Controlling the Order of Execution" section).

Referencing a Procedure in a World or Directory

A _Start file with the following contents illustrates the basic set of annotations required to reference a procedure that resides in an Environment world or directory:

```
--|Procedure_Name Initialize_Routine
--|Procedure_Context !Commands.Example
--|Parameters Notify => "manager",
--|Parameters Effort_Only => False
```

- The `--|Procedure_Name` annotation specifies the name of the referenced procedure. If the procedure resides directly in a library, you supply the procedure's simple name; if it resides in a package, supply the name in the form *Package_Name.Procedure_Name*.
- The `--|Procedure_Context` annotation specifies the Environment world or directory that contains the referenced procedure.
- Each of the `--|Parameters` annotations specifies the value to be used for one of the procedure's parameters. Note that string values must be enclosed in quotation marks, and commas must be included to separate multiple parameters.

Referencing a Procedure in a Subsystem

A _Start file with the following contents illustrates how to reference a procedure that resides in an Environment subsystem. You must replace the `--|Procedure_Context` annotation with the `--|Subsystem` and (optional) `--|Activity` annotations:

```
--|Procedure_Name Excelan_Boot_Server
--|Subsystem !Targets.Implementation.Motorola_68k_Download
--|Activity !Machine.Release.Current.Activity
--|No_Wait
```

- The `--|Subsystem` annotation specifies the name of the subsystem that contains the procedure. (When this annotation is specified, the `--|Procedure_Context` annotation is ignored.)
- The `--|Activity` annotation specifies the activity to be used to obtain the view name and construct the full pathname of the procedure. If you omit this annotation, !Machine.Release.Current.Activity is used.
- Note that Excelan_Boot_Server is a parameterless procedure; otherwise, one or more `--|Parameters` annotations would be present.
- The `--|No_Wait` annotation permits concurrent execution (see the "Controlling the Order of Execution" section). This annotation is present because this procedure starts a server.

Specifying Further Information

_Start files may also contain annotations that control the order in which the Start procedure will execute the referenced procedures. See the "Controlling the Order of Execution" section.

Using a _Start file is equivalent to executing the referenced procedure through Program.Run_Job or Program.Run. See the comments in the specification of !Machine.Initialization.Start for additional annotations that specify the equivalent of the Options parameter in the Program.Run_Job procedure and the Context parameter in the Program.Run and Program.Run_Job procedures.

Controlling the Order of Execution

You can specify the relative execution order of all initialization procedures, including those referenced in _Start files. To do this, you include annotations in the appropriate procedure specifications or _Start files:

- To specify that procedure A cannot run until procedure B has finished, you include the annotation `--|Prerequisite B` in the specification of procedure A (or in the _Start file that references procedure A).

  If procedure B is referenced in a _Start file, you specify the filename as the annotation's argument: `--|Prerequisite B_Start`.

  Note that the argument of this annotation is a simple name and that all three worlds (Rational, Local, and Site) are searched for that simple name. Therefore, simple names must be unique across these three worlds if you want to use this annotation.
- To specify that procedure A must finish before any other procedure can start executing, you include the annotation `--|Wait` in the specification of procedure A (or in the _Start file that references procedure A).

  Using this annotation is equivalent to executing procedure A using the Program.Run procedure.
- To specify that procedure A is to execute as a separate job concurrent with other procedures, you include the annotation `--|No_Wait` in the specification of procedure A (or in the _Start file that references procedure A).

  Using this annotation is equivalent to executing procedure A using the Program.Run_Job procedure.

If none of the annotations listed above are present in a given procedure or _Start file, the `--|Wait` annotation is assumed. That is, procedures are executed sequentially unless told otherwise.

If a circular dependency results from a combination of annotations, it will be reported and ignored, so that each procedure will run.

Note that you can execute the Start command with the Effort_Only parameter set to True to test the execution order that results from your annotations.

See the comments in the specification of the !Machine.Initialization.Start procedure for a complete description of annotation usage, along with examples.

Customizing Disk-Collection Thresholds

You can customize the disk-collection thresholds for the particular needs of your site. To implement a change in your disk-collection thresholds:

1. Create an empty Ada unit in the Local world.
2. Copy the contents of the !Machine.Initialization.Local.Local_Gc_Thresholds-_Sample file into the empty Ada unit.
3. In the Ada unit, edit the Thresholds1 and Thresholds2 arrays to specify the desired thresholds.
4. Promote the Ada unit.

Note, however, that values set by the procedure in the Rational world are calculated based on your disk configuration and should be sufficient; see your Rational technical representative if you want to use different thresholds.

ENABLING AND CONFIGURING LOGIN PORTS

D_12_5_0 and subsequent releases provide a file-driven mechanism in !Machine-.Initialization for enabling and configuring ports for login.

At the very least, you must ensure that this mechanism has enough information to enable the desired login ports (see the "Enabling Ports for Login" section). In addition, you may optionally use this mechanism to specify:

- Connection and terminal-type characteristics for Telnet and RS232 ports, such as logoff on disconnect, disconnect on logoff, and so on
- Communication characteristics for RS232 ports, such as flow control, parity, and so on

Such information is specified using the options described in the "Terminal-Configuration Options" section.

Enabling Ports for Login

Ports for terminal devices must be enabled for login each time an R1000 boots. Accordingly, the !Machine.Initialization.Start procedure calls a procedure called Terminals in the Rational world. This procedure in turn consults a file called Terminal_Configuration in the Local world to determine which ports to enable for login. This file-driven mechanism takes the place of a procedure such as !Machine.Initialize_Terminals, which enables terminals through calls to the Operator.Enable_Terminal procedure.

The Terminal_Configuration file is created automatically in the Local world during installation:

- On a new machine, a Terminal_Configuration file is created that enables ports 235 through 249 for login.
- On an R1000 that is being upgraded from a previous Environment release, the Terminal_Configuration file contains entries for the same set of ports that were enabled at the time of installation. (This file also preserves any nondefault communication characteristics that were in effect for RS232 ports; see the "Terminal-Configuration Options" section.)

You can edit the Terminal_Configuration file at any time to change which ports are enabled. The changes take effect the next time you boot the R1000. Alternatively, you can execute the Rational.Terminals procedure to make the changes take

effect without booting the R1000. Note that you must keep the Terminal_Configuration file in the Local world, even if you want to enable the same ports on all machines at a given site.

Following is a sample Terminal_Configuration file that contains basic enabling information:

```
16 => (Enable)
224 .. 249 => (Enable)
-- Ports 250 and 251 are for printers; disable them for login
250..251 => (Login_Disabled)
```

As shown, the Terminal_Configuration file consists of:

- Comments preceded with Ada comment notation (--)
- Entries of the general form: *Port_Range* => (*Options*), where:
  - *Port_Range* can be a single port number or a range of port numbers
  - *Options* must be enclosed in parentheses

The options that pertain to enabling and disabling ports are listed in Table E-1.

Table E-1 Options for Enabling and Disabling Ports

Option	Description
Enable	When specified for a given port, enables the port for login. Note that the port cannot subsequently be enabled for any other device, such as a printer.
Login_Disabled	When specified for a given port, prevents the port from being enabled for login—for example, by subsequent usage of the Operator.Enable_Terminal command. Note that the port can subsequently be enabled for other devices, such as printers.
Disable	When specified for a given port, disables the port for all devices. Note that the port can subsequently be enabled for any device, including login. Specifying this option is equivalent to having no entry for the port in the file.

Port 16 is always enabled for login, regardless of whether an entry exists for it. An entry for port 16 is included in the automatically created Terminal_Configuration file for explicitness.

Do not assign the Enable option to any port that you plan to enable for a printer or other device (such as a CDF). Instead, you can assign the Login_Disabled option or the Disable option to those ports, or you can simply omit entries for them from the file. Assigning the Login_Disabled option is recommended if you want to ensure that printer ports cannot be enabled for login even if the print spooler is killed.

Customizing Port Characteristics

You can add information to the Terminal_Configuration file to specify connection characteristics for RS232 ports and communication characteristics for RS232 and Telnet ports. Such information is specified through the options listed in the "Terminal-Configuration Options" section.

The simplest way to specify multiple options is to assign them directly to a port or range of ports:

- Multiple options in a single entry must be enclosed in parentheses.
- Multiple options must be separated by commas.
- The options can extend over several lines, although the entry itself must start on a new line.

For example, the following entry assigns several connection characteristics to ports 224..249 and then enables those ports:

```
224..249 => (Logoff_On_Disconnect, Disconnect_On_Logoff, Enable)
```

You can organize recurrent sets of options and improve readability in the Terminal_Configuration file by defining an abbreviation for each set of options and then assigning each abbreviation to a port or range of ports:

- Abbreviation entries are of the general form *Abbreviation = Options*. Note that the equals sign (=) is used to define abbreviations; the => symbol is used for port assignment.
- Existing abbreviations can be nested in the definition of new abbreviations.

For example, the following entries create the abbreviations User_Ports and Telnet_Ports, assigning the Telnet_Ports abbreviation to ports 224..249:

```
-- Port settings for user login ports
User_Ports   = (Logoff_On_Disconnect, Disconnect_On_Logoff)

-- Port settings for Telnet ports
Telnet_Ports = (Terminal_Type => Xrterm, User_Ports)

224..249     => (Telnet_Ports, Enable)
```

When adding entries to a Terminal_Configuration file, bear in mind that:

- Nondefault communication characteristics for RS232 ports must be set each time an R1000 boots. Consequently, if a port is to have nondefault values for any of the options listed in Table E-4, you must include these options in the entry for that port. Omitting an option causes its default value to be set.
- Connection and terminal-type characteristics persist across boots, retaining the last values that were set for them. Thus, in principle, the options listed in Tables E-2 and E-3 need to be set only once and then can be omitted from the Terminal_Configuration file. However, you may choose to include values for these options in the file to ensure that booting the system resets them to the proper values in case they had been changed.
- The options for each port are set in the order in which they are assigned in the Terminal_Configuration file. Similarly, the options in an abbreviation are set in the order in which they are declared. If a single port number is included in the ranges of more than one entry, that port takes the options of the last entry in which it appears.

A Sample Terminal_Configuration File

The following sample file shows how a system manager can use abbreviations to organize port information meaningfully. Note that a number of connection options have been explicitly set to ensure that booting the system sets them to a known value. Note also that specifying the Disable option for the printer ports is not absolutely necessary; however, specifying this option ensures that no previous entry in the file had inadvertently enabled these ports.

```
-- Operator line 16 settings
Operator_Port = (~Logoff_On_Disconnect,
                 ~Disconnect_On_Logoff,
                 ~Login_Disabled)
-- User login port settings
User_Ports = (Logoff_On_Disconnect, Disconnect_On_Logoff,
              ~Login_Disabled, ~Log_Failed_Logins,
              ~Disconnect_On_Failed_Login,
              ~Disconnect_On_Disconnect)

-- Dial-in port connection settings
Dialin_Ports = (Terminal_Type => VT100,
                Input_Rate => Baud_2400, Output_Rate => Baud_2400,
                Parity => None, Bits_Per_Char => Char_8,
                Stop_Bits => 1, User_Ports)

-- Telnet port settings
Telnet_Ports = (Terminal_Type => Xrterm, User_Ports)

-- Printer port settings
Printer_Ports = (Login_Disabled)

-- Ports not in use
Unused = (Login_Disabled)

16 => (Operator_Port, Enable)
17..31 => (Dialin_Ports, Enable)
224..249 => (Telnet_Ports, Enable)
250..251 => (Disable, Printer_Ports)
252..255 => (Disable, Unused)
```

Terminal-Configuration Options

Tables E-2, E-3, and E-4 summarize the connection, terminal type, and RS232 communication options you can specify in the Terminal_Configuration file. These options invoke corresponding procedures in package Terminal.

Table E-2 Boolean Options for Connection Characteristics

Option	Description
`Disconnect_On_Disconnect`	When specified for a given port, causes the Environment to respond to an incoming disconnect signal received on that port by initiating an outgoing disconnect signal on that port.
`Disconnect_On_Failed_Login`	When specified for a given port, causes the Environment to initiate an outgoing disconnect signal on that port when a user repeatedly fails to log in on that port (for example, by repeatedly entering an incorrect password or unrecognized username).
`Disconnect_On_Logoff`	When specified for a given port, causes the Environment to initiate an outgoing disconnect signal on that port when a user logs off a session running on that port.
`Log_Failed_Logins`	When specified for a given port, causes the Environment to write an entry to the system error log when a user fails repeatedly to log in on that port (for example, by repeatedly entering an incorrect password or unrecognized username).
`Logoff_On_Disconnect`	When specified for a given port, causes the Environment to respond to a disconnect received on that port by logging off that port's session.

Table E-3 Enumeration Option for Specifying Terminal Type

Option => *Value*	Description
`Terminal_Type`	Specifies the output driver type for a given port. *Value* can be any valid terminal type name, including (but not limited to): `Cit500r Facit Rational Vt100 Xrterm`

Table E-4 Enumeration Options for RS232 Communication Characteristics

Option => *Value*	Default Value	Description
`Bits_Per_Char`	`Char_8`	Specifies the number of data bits per character. *Value* can be: `Char_5 Char_6 Char_7 Char_8`
`Flow_Control`	`None`	Specifies software flow control for data transmitted by the R1000 on the specified port. *Value* can be: `None Xon_Xoff Dtr Rts`

`Input_Rate`	`Baud_9600`	Sets the incoming data rate for a given port. *Value* can be:
		`Baud_50 Baud_75 Baud_110`
		`Baud_134_5 Baud_150 Baud_200`
		`Baud_300 Baud_600 Baud_1200`
		`Baud_1800 Baud_2400 Baud_9600`
		`Baud_19200 Disabled Ext_Rec_Clk`

`Output_Rate`	`Baud_9600`	Sets the outgoing data rate for a given port. Values are the same as for Input_Rate.
`Parity`	`None`	Sets the parity for transmitted and received data on a given port. *Value* can be: `None Odd Even`
`Stop_Bits`	`2`	Sets the number of stop bits for a given port. *Value* is a natural number in the range 1..2.
`Receive_Flow_Control`	`None`	Specifies flow control of data received by the R1000 on the specified port. *Value* can be: `None Xon_Xoff Dtr Rts`

Table E-4 Enumeration Options for RS232 Communication Characteristics (continued)

Option => *Value*	Default Value	Description
`Receive_Xon_Xoff_Bytes`	`(17,19)`	Specifies flow-control bytes so that the R1000 can regulate the data it receives on the specified port. *Value* is `(n,m)`, where *n* and *m* are natural numbers in the range 0..255.
`Xon_Xoff_Bytes`	`(17,19)`	Specifies the flow-control bytes that the R1000 recognizes for the specified port. *Value* is: `(n,m)`, where *n* and *m* are natural numbers in the range 0..255.

CONFIGURING PRINTERS

D_12_5_0 and subsequent releases provide a file-driven mechanism in !Machine.Initialization for configuring a group of networked and/or local printers. This mechanism allows you to define a *printer name* for each printer on the network and to specify how each printer is to be accessed. Furthermore, you can associate a printer name with each user, so that when a given user enters the Print command (that is, !Commands.Abbreviations.Print), the print job will be sent by default to the device that is defined by the associated printer name.

This file-driven mechanism automatically adds the specified devices to the appropriate R1000s, creates the necessary print classes on the appropriate R1000s, and associates each class with the specified device, thereby creating print queues. Thus, when you use the mechanism, you do not need to use procedures from package Queue (such as Add, Create, Enable, and Register) to do these tasks.

Existing sites can choose whether to use this file-driven mechanism or to continue using procedures from package Queue to configure printers. However, because the file-driven mechanism combines class and machine information, it is recommended that large sites with multiple networked printers use the file-driven !Machine.Initialization mechanism. Small sites with few printers connected directly to R1000s may want to continue using package Queue. Sites that want to create a single class with multiple devices should also use package Queue.

Note that the printer configuration (set through either commands in package Queue or the file-driven mechanism) applies to both the Queue.Print and Abbreviations.Print commands. The ability to associate printer names with usernames is specific to the Abbreviations.Print command.

Where to Specify Printer Information

Each time an R1000 boots, the !Machine.Initialization.Start procedure calls a procedure called Printers in the Rational world. This procedure initializes the print spooler on that R1000 based on the information in the following user-created files:

- !Machine.Initialization.Site.Printer_Configuration, which defines a printer name for each of the devices available on the network and specifies how each device is to be accessed. A copy of this file must exist on all R1000s from which users will enter the Print command and on all R1000s that will handle print requests for the specified devices.

- !Machine.Initialization.Local.Printer_Configuration, which defines additional printer names for additional devices intended only for users of the current R1000.
- !Machine.Initialization.Local.User_Printer_Map, which associates a default printer name with individual users on the current R1000. When a user executes the Print command with the default Printer parameter, the command looks up the user's name and sends the print request to the corresponding printer.

At a minimum, the Print command requires that one Printer_Configuration file exist in either the Site or Local world. If no User_Printer_Map file exists (or if the file exists but contains no entry for a particular user), the Print command uses the first printer name defined in the Site.Printer_Configuration file. If this file does not exist, the first printer name defined in the Local.Printer_Configuration file is used.

The classes and devices specified in the Printer_Configuration files are created when the Printers procedure (that is, !Machine.Initialization.Rational.Printers) is executed (normally, during machine initialization). The Printers procedure also associates usernames with the appropriate printer information. Be aware that whenever information is changed in any of the files listed above, the Printers procedure must be run in order to update the information.

If you choose not to create any of these files, you will have to:

- Create an initialization procedure (in either the Site or Local world) that uses package Queue to create print queues each time the system boots.
- Either:
  - Use the Queue.Print command instead of the !Commands.Abbreviations.Print command.
  - Use the !Commands.Abbreviations.Print command, but explicitly specify the class name for the Printer parameter.

Adding Entries to a Printer_Configuration File

During installation, an empty Printer_Configuration file is created in the Local world. After installation, you can:

- Add entries to this file to enable the use of the Print command.
- Move this file (or create an additional file called Printer_Configuration) in the Site world, if you need to define sitewide printer information.

Each entry in a Printer_Configuration file defines a printer name and specifies the characteristics of the device it represents.

Each entry must start on a new line, but the information can extend over several lines and can include single and in-line comments. For readability, the entries are often formatted like command parameters.

Each entry has the general form:

```
Printer_Name => (Device_Type  => Device_Info,
                                 Other_Device_Characteristics,
        Laser_Comm            => Boolean,
        Reverse_Output_Pages  => Boolean,
        On_Node               => R1000_Name)
```

where:

- *Printer_Name* is the name by which you want to refer to a given device in a Print command. *Printer_Name* must be a legal Ada simple name.
- *Device_Type* is one of the following four kinds of printer devices:
  - Direct, which specifies a printer connected to an R1000 via direct line.
  - Telnet, which specifies a printer connected to an R1000 via Telnet.
  - File, which specifies a file on an R1000 in which print-spooler output is collected. Subsequent processing is required to get this output printed.
  - Workstation, which specifies a directory on a remote workstation to which print requests are sent. Such requests are sent via FTP as individual files; from the remote directory, they can be printed using the workstation's print tools.
- *Device_Info* is further information about the device you specified. The information depends on the type of device specified (see the paragraphs below).
- *Other_Device_Characteristics* are additional entry elements, separated by commas, that give further information about the chosen device (see the paragraphs below).
- The Laser_Comm option, when True, specifies that printing will be done on a laser printer. Omitting this option implies that printing will be done on a line printer.

  For Direct and Telnet devices, setting Laser_Comm to True specifies that a laser printer is connected; for File and Workstation devices, setting Laser_Comm to True specifies that the collected print requests will eventually be printed on a laser printer.
- The Reverse_Output_Pages option allows you to adjust the order in which pages are spooled to accommodate the way your printer stacks pages in its output tray. This option applies only if the Laser_Comm option is set to True.
  - Setting Reverse_Output_Pages to True causes the print spooler to reverse the order of output pages, so that the last logical page is printed first. Omitting this option is equivalent to specifying True.
  - Setting Reverse_Output_Pages to False causes the print spooler to keep the pages of output in the order in which they appear in the source file.
- The On_Node option specifies the network name of the R1000 that contains the print spooler for the device. Omitting this option is equivalent to specifying the name of the current R1000.

The following paragraphs describe printer-configuration file entries for each of the four kinds of devices. (See also the comments in the specification of !Machine-.Initialization.Rational.Printers.)

Specifying a Directly Connected Printer

To specify a printer connected to an R1000 via direct line, you specify an entry of the following general form:

```
Printer_Name => (Direct                => Protocol,
                 Device                => Terminal_N,
                 Laser_Comm            => Boolean,
                 Reverse_Output_Pages  => Boolean,
                 On_Node               => R1000_Name)
```

where:

- *Protocol* specifies the printer flow control and can be either Xon_Xoff or Dtr. See the printer manual for details.
- Terminal_*N* represents the RS232C port to which the printer is connected (*N* is the port number). The specified port must not be enabled for login in the Local.Terminal_Configuration file.
- The Laser_Comm option, when True, specifies that a laser printer is connected and enables a two-way printer-communication protocol. Omitting the option is equivalent to specifying False, which means that a line printer is connected.
- The Reverse_Output_Pages option allows you to adjust the order in which pages are spooled to accommodate the way your printer stacks pages in its output tray, as described in the "Adding Entries to a Printer_Configuration File" section. This option applies only if Laser_Comm is set to True.
- The On_Node option specifies the network name of the R1000 to which the printer is directly connected. Omitting this option is equivalent to specifying the name of the current R1000.

The following entry creates a printer name called Lp, which represents a line printer that is directly connected to port 30 of an R1000 called Jazmo:

```
-- Line printer connected to Jazmo
Lp => (Direct  => Xon_Xoff,
       Device  => Terminal_30,
       On_Node => Jazmo)
```

Specifying a Networked Printer

To specify a printer connected to an R1000 via Telnet, you specify an entry of the following general form:

```
Printer_Name => (Telnet                => Host_Name,
                 Device                => Terminal_N,
                 Laser_Comm            => Boolean,
                 Reverse_Output_Pages  => Boolean,
                 On_Node               => R1000_Name)
```

where:

- *Host_Name* is the network name of the printer.
- Terminal_*N* represents the Telnet port to which the printer is connected (*N* is the port number). The specified port must not be enabled for login in the Local.Terminal_Configuration file.
- The Laser_Comm option, when True, specifies that a laser printer is connected and enables a two-way printer-communication protocol. Omitting the option is equivalent to specifying False, which means that a line printer is connected.
- The Reverse_Output_Pages option allows you to adjust the order in which pages are spooled to accommodate the way your printer stacks pages in its output tray, as described in the "Adding Entries to a Printer_Configuration File" section. This option applies only if Laser_Comm is set to True.
- The On_Node option specifies the network name of the R1000 to which the printer is connected via Telnet. Omitting this option is equivalent to specifying the name of the current R1000.

The following entry creates the printer name Dlaser, which represents a laser printer that is connected via Telnet to port 226 of an R1000 called Roget. Because of the way this printer stacks its output, print requests are spooled to this device with their pages in ascending (rather than reversed) order:

```
-- Documentation's laser printer
Dlaser => (Telnet                => Doc_Laser,
           Device                => Terminal_226,
           Laser_Comm,
           Reverse_Output_Pages => False,
           On_Node               => Roget)
```

Specifying an Environment File

To specify a file in which to collect print-spooler output, you specify an entry of the following general form:

```
Printer_Name => (File                 => Environment_Pathname,
                 Laser_Comm           => Boolean,
                 Reverse_Output_Pages => Boolean,
                 On_Node              => R1000_Name)
```

where:

- *Environment_Pathname* specifies the file to which output is written. The pathname must name a file that exists on the R1000 named by On_Node. Note that the group Spooler must have access to the specified file.
- The `Laser_Comm` option, when True, specifies that the collected print requests will eventually be printed on a laser printer. Omitting the option is equivalent to specifying False, which means that a line printer will be used.
- The `Reverse_Output_Pages` option allows you to adjust the order in which pages are spooled to accommodate the way your printer stacks pages in its output tray, as described in the "Adding Entries to a Printer_Configuration File" section. This option applies only if Laser_Comm is set to True.
- The `On_Node` option specifies the network name of the R1000 on which the file is located. Omitting this option is equivalent to specifying the name of the current R1000.

The following entry creates the printer name Hold, which represents a file on an R1000 called Logo. Low-priority print requests are sent to this file, where they are held until someone prints the file using the Print command (specifying a printer name that represents a connected printer):

```
-- Place to hold large requests until printers are free in the
-- evening
Hold => (File    => !Machine.Queues.Local.Held_Print_Requests,
         On_Node => Logo)
```

Specifying a Workstation Directory

To specify a directory on a workstation to which print requests are sent, you specify an entry of the following general form:

```
Printer_Name => (Workstation          => Host_Name,
                 Path                 => Directory_Name,
                 Laser_Comm           => Boolean,
                 Suffix               => String,
                 Reverse_Output_Pages => Boolean,
                 On_Node              => R1000_Name)
```

where:

- *Host_Name* is the network name of the workstation to which the files will be transferred.
- *Directory_Name* is the pathname of the workstation directory into which the files will be transferred. The directory pathname must have syntax appropriate to the workstation and must have trailing punctuation that permits the name of the transferred print-request file to be appended.
- The `Laser_Comm` option, when True, specifies that the collected print requests will eventually be printed on a laser printer. Omitting the option is equivalent to specifying False, which means that a line printer will be used.
- The `Suffix` option allows you to specify a string to be appended to the filenames that are created on the workstation. Omitting this option causes no suffix to be appended to the filenames. The specified suffix can be used by print tools as a way of identifying which files to print. This is useful when several printer names send files to the same directory.
- The `Reverse_Output_Pages` option allows you to adjust the order in which pages are spooled to accommodate the way your printer stacks pages in its output tray, as described in the "Adding Entries to a Printer_Configuration File" section. This option applies only if Laser_Comm is set to True.
- The `On_Node` option specifies the network name of the R1000 whose print spooler handles the print requests. Omitting this option is equivalent to specifying the name of the current R1000.

In addition to creating an entry for each workstation in the Printer_Configuration files, you must also create a remote-passwords file, containing the username and password to be used for accessing each workstation. This remote-passwords file must be named Remote_Access and exist in the library !Machine.Queues.Ftp. (For information about remote-passwords files, see package Remote_Passwords in the Session and Job Management (SJM) book of the *Rational Environment Reference Manual.*)

When print requests are sent as files to a workstation, any FTP messages are directed to a log file that is created in !Machine.Queues.Ftp. This log file is automatically cleared after each 100 print requests. Messages pertaining to creating print classes and enabling devices are directed to the system error log.

The following two entries create printer names Dc_Laser and Dc_Lineprinter, both of which direct print requests to a directory on a UNIX workstation called Enterprise. These print requests, which are routed through the print spooler on an R1000 called Capitol, are sent as files with different suffixes, depending on the printer name (`_Lsr` and `_Lpt`, respectively):

```
-- Laser printer attached to workstation in Washington, D.C.,
-- office; spooled on R1000 called Capitol.
Dc_Laser => (Workstation => Enterprise,
             Path        => /usr/spool/ratqueue/,
             Laser_Comm,
             Suffix      => _Lsr,
             On_Node     => Capitol)
```

```
-- Line printer attached to workstation in Washington, D.C.,
-- office; spooled on R1000 called Capitol.
Dc_Lineprinter => (Workstation => Enterprise,
                   Path        => /usr/spool/ratqueue/,
                   Suffix      => _Lpt,
                   On_Node     => Capitol)
```

Print tools on the workstation, such as the following sample, are required to actually print the requests:

```
# This program spools print requests placed in /usr/spool/ratqueue to
# the appropriate printer based on the suffix of the spooled file.
# It checks the spool directory at 5-minute intervals to see if any
# new files need printing. This runs as a C-shell script. To execute,
# type csh and the name of this file.

:
set xFTPxDIRx=/usr/spool/ratqueue              #spool directory
#
set xFTPxSUFFIXxLSRx=_Lsr       # Laser printer suffix
set xFTPxSUFFIXxLPTx=_Lpt       # Line printer suffix
#
set xPRINTxLISTxLSRx=/tmp/rat_print_lsr.$$     # Laser printer list file
set xPRINTxLISTxLPTx=/tmp/rat_print_lpt.$$     # Line printer list file
#
set xPRINTxCOMMANDx=/tmp/rat_command.$$        # temporary print command file
#
set xPRINTxDELAYx=120           # interval to wait before printing
set xRECHECKxTIMEx=180          # interval for checking print requests
#
cd $xFTPxDIRx
while (1 == 1)
#
# Creates a list of files to print for each printer.
#
    ls $xFTPxDIRx | grep $xFTPxSUFFIXxLSRx > $xPRINTxLISTxLSRx
    ls $xFTPxDIRx | grep $xFTPxSUFFIXxLPTx > $xPRINTxLISTxLPTx
#
# Adds the laser- and line-printer files to the print-command file
# (the device name of each type of printer should be provided after the -P).
#
    cat $xPRINTxLISTxLSRx | sed -e 's^.^lpr -r -Plaser &^' > $xPRINTxCOMMANDx
    cat $xPRINTxLISTxLPTx | sed -e 's^.^lpr -r -Pline &^' >> $xPRINTxCOMMANDx
#
# Waits the specified amount of time before printing the requests.
# (This delay allows time for the FTP operation to complete and should
# be adjusted based on the average length of files being printed.)
#
    sleep $xPRINTxDELAYx
#
# Prints the files if there are any to print.
#
    set LCNT=`cat $xPRINTxCOMMANDx | wc -l `
    if ( $LCNT != 0 ) then
        chmod +x $xPRINTxCOMMANDx
        $xPRINTxCOMMANDx
    endif
#
# Waits the specified amount of time.
#
```

```
    sleep $xRECHECKxTIMEx
#
# Removes the old temporary files.
#
    rm $xPRINTxLISTxLSRx
    rm $xPRINTxLISTxLPTx
    rm $xPRINTxCOMMANDx
#
end
```

Associating Default Printers with Individual Users

To associate a default printer with each user, you must create a file called User_Printer_Map in the Local world and add entries of the following form:

Username Printer_Name

where:

- *Username* is either:
  - The username for an R1000 user
  - The string `others`, which represents all users not explicitly listed by name.
- *Printer_Name* is defined in a Printer_Configuration file.

Following is a sample User_Printer_Map file:

```
phil dc_laser
sue dlaser
others lp
```

See also the comments in the specification of !Machine.Initialization.Rational_Printers.

F Quick Reference for Parameter-Value Conventions

Parameters of Rational Environment commands accept values that conform both to Ada and to Environment-defined conventions. This Quick Reference summarizes the Environment-defined conventions for parameter values.

WHERE TO LOOK

- Conventions for referencing objects:

- Conventions for specifying other command inputs:

- Explanations and examples of all Environment-defined parameter-value conventions:

  Parameter-Value Conventions tabbed section in the Reference Summary (RS) book of the *Rational Environment Reference Manual*

- Introductory material describing Ada parameter syntax:

  Rational Environment User's Guide

PATHNAMES

- Must be enclosed in quotation marks.
- Consist of name components separated by periods.
  - *Fully qualified pathnames* start with !:
    `"!Users.Anderson.Calculation"`
  - *Relative pathnames* start with a character other than ! and are resolved relative to the current library:
    `"Anderson.Calculation"`
  - *Simple names* contain only one component:
    `"Calculation"`
  - Pathnames of *deleted* objects are enclosed in braces:
    `"{Old_Memo}"`

LIBRARY.RESOLVE COMMAND

- Is useful for testing naming expressions before you use them in commands.
- Evaluates special names, context characters, wildcard characters, substitution characters, set notation, indirect file notation, and attributes.
- Displays the fully qualified pathname(s) to which an expression expands, relative to the context in which the command is entered:

```
Library.Resolve (Name_Of => "[@,~lee,~miyata,~chavez]");
```

- Displays the fully qualified pathname(s) to which a pair of source and destination expressions expand (as when used in move/copy operations):

```
Library.Resolve (Name_Of     => "File@",
                 Target_Name => "Old_File@");
```

DESIGNATION

There are three ways to designate an object (specify it for command operation):

- Position the cursor on its name or in its image.
- Select (highlight) its name or image using commands from packages !Commands.Common.Object or !Commands.Editor.Region.
  - Useful key combinations: [Object] - [←]
    [Region] - [[] and [Region] - []]
- Select its name or image *and* position the cursor in the highlighted area.

PARAMETER PLACEHOLDERS

- Appear as default parameter values in many commands.
- Have the form `">>clue<<"` (for example, `">>LIBRARY NAME<<"`).
- Must be replaced with a pathname appropriate to the *clue*.

SPECIAL NAMES

- Are predefined strings that provide a shorthand for referencing various objects.
- Must be enclosed in quotation marks.
- Can be upper- or lowercase.
- Can be abbreviated to shortest unambiguous string within the angle brackets.

Special Names for Specific Objects

Shortest Form	Full Form	Description
"<h>"	"<HOME>"	Resolves to the user's home world.
"<su>"	"<SUBSYSTEM>"	Resolves to the enclosing subsystem.
"<vi>"	"<VIEW>"	Resolves to the enclosing view.

Special Names for Designated Objects

Shortest Form	Full Form	Description
"<c>"	"<CURSOR>"	Resolves to the object on which the cursor is located; any highlighted area is ignored.
"<r>"	"<REGION>"	Resolves to the highlighted object; cursor can be anywhere. Often specifies a highlighted source object in a copy command, where cursor specifies the destination.
"<s>"	"<SELECTION>"	Resolves to the highlighted object; the highlight must contain the cursor or an error results.
"<i>"	"<IMAGE>"	Resolves to the highlighted object containing the cursor (if any); otherwise, resolves to the object whose image contains the cursor.
"<t>"	"<TEXT>"	Resolves to the object whose name is a highlighted string containing the cursor. Equivalent to copying the highlighted string directly into the parameter prompt.

Special Names for Default Objects

Shortest Form	Full Form	Description
"<a>"	"<ACTIVITY>"	Resolves to the highlighted activity containing the cursor (if any); otherwise, resolves to the default activity for the current session.
"<sw>"	"<SWITCH>"	Resolves to the highlighted library switch file; otherwise, resolves to the library switch file associated with the highlighted library; otherwise, resolves to the library switch file associated with the current image. The highlighted area (if any) must contain the cursor.

CONTEXT CHARACTERS

- Are shorthand for object names based on object location or relationship to other known objects.

Context Characters

Character	Description
!	Resolves to the Environment's root world.
!!*name*	Resolves to the Environment's root world on an R1000 called *name* (only for commands in packages Archive and Queue).
[]	Resolves to the current context (either a library or an Ada unit).
$	Resolves to the current library, when used alone or at the beginning of a name. (The current library either is or encloses the current context.)
$$	Resolves to the current world, when used alone or at the beginning of a name. (The current world is a world, view, or subsystem that either is or encloses the current context.)
^	Resolves to the closest enclosing object (either a library or an Ada unit), when used alone or at the beginning of a name.
$[*name*]	Resolves to the closest enclosing library whose simple name is *name*.
$$[*name*]	Resolves to the closest enclosing world whose simple name is *name*.
^[*name*]	Resolves to the closest enclosing object whose simple name is *name*.
foo	Evaluates the name *foo* relative to the searchlist for the current session.
library`*foo*	Evaluates the simple name *foo* relative to the links associated with *library*.
unit`*foo*	Evaluates the simple name *foo* relative to the objects that are directly visible in *unit*.

DEBUGGER CONTEXT CHARACTERS

- Are used in debugger commands to reference stack frames, tasks, and Ada units in the program being debugged.
- Can be used in addition to general context characters (see above).

Debugger Context Characters

Character	Description
.*name*	Resolves *name* to an Ada unit that is referenced in the program being debugged. Subsequent name components resolve to objects declared in that Ada unit: `Source (".Initialize.Status")`
_*n*	Refers to the stack frame with the specified frame number (1 is the top frame). Subsequent name components resolve to objects in the subprogram whose activation is contained in the frame: `Put ("_3.Status")`
%*name*, %*n*	Refers to the task with the specified task name or number. Subsequent name components resolve to objects declared in the task: `Put ("%Debug_Shell._3.Status")`

WILDCARD CHARACTERS

- Match portions of pathnames.
- Allow you to abbreviate single pathnames or specify multiple pathnames as a single string.
- Differ from *restricted naming expressions* and *pattern-matching characters* for search strings (see page 365).

Wildcard Characters

Character	Description
#	Matches a single character: `T####` matches Tools.
@	Matches zero or more characters: `!U@.@.Tools` matches !Users.Anderson.Tools.
?	Matches zero or more nonworld name components: `!Users.Anderson?` matches !Users.Anderson and all objects in it except worlds.
??	Matches zero or more name components, including worlds and their contents: `!Users??` matches !Users, all user home worlds, and their contents.

SUBSTITUTION CHARACTERS

- Are shorthand for specifying destination names in move and copy operations.
- Derive one or more destination names from portions of source names.

Substitution Characters

Character	Description

@	Expands to the string or strings matched by a wildcard (#, @, ?, ??) in the source name; allows you to copy, move, or rename multiple objects in a single operation. The following command renames File1 through File50 to be Old_File1 through Old_File50:
	`Lib.Rename (From => "File@",  -- wildcard @`
	`To   => "Old_File@"); -- substitution @`

#	Expands to a name component from the source name; saves typing when the source and destination names have components in common. The following command copies !Users.Anderson.Memo to !Users.Anderson.Documentation.Memo:
	`Lib.Copy (From => "!Users.Anderson.Memo",`
	`To   => "!#.#.Documentation.#");`

SET NOTATION

- Allows you to specify multiple names in a single string.

Set Notation

Notation	Description

`[...]`	Delimits a set of fully qualified or relative names:
	`"[!users.lee,!users.miyata,!users.chavez]"`
	or name segments:
	`"!users[lee,miyata,chavez]tools"`

`[...`*`name;name`*`...]`	Separates names in set; requires every name in the set to resolve (an unresolved name is an error): `"[lee;miyata;chavez]"`
`[...`*`name,name`*`...]`	Separates names in set; allows unresolved names to be ignored without causing errors: `"[lee,miyata,chavez]"`
`[...~`*`name`*`...]`	Excludes a name from a set: `"!users[@,~lee,~miyata,~chavez]"`

INDIRECT FILES

- Provide a convenient way to use the same set of pathnames in multiple commands or in commands that are entered multiple times.
- Are specified as a parameter value by prefixing the filename with an underscore:

```
Archive.Save (Objects => "_users_in_my_group");
```

- Contain a list of fully qualified or relative pathnames; can contain *wildcards*, *attributes*, *set notation*, and other indirect files.
  - Names are entered on separate lines or are separated by commas or semicolons (see separators for *set notation*).
  - Names on separate lines are the same as names separated by commas:

```
!users.lee
!users.miyata
!users.chavez
```

  - Command converts contents of indirect file into set notation.

RESTRICTED NAMING EXPRESSIONS

- Are used in parameters that accept only a restricted subset of naming possibilities (the For_Prefix of Archive.Copy, the Source parameter of Links.Display).
  - Such parameters match string names against a list rather than resolving them against the library system.

Restricted Naming Expressions

Character	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 name components of any kind: `!Users.Anderson?` matches !Users.Anderson and everything in it.
`[...]`	Encloses a set of names: `[!Users.Anderson?,!Users.Miyata?]` matches everything in the home worlds for Anderson and Miyata.
`~name`	Excludes a name from a set: `[@,~Tools]` matches everything except Tools.

PATTERN-MATCHING CHARACTERS

- Are used in regular expression matching for search and comparison commands (packages Editor.Search and File_Utilities).

Wildcard Characters for Pattern Matching

Character	Description
`?`	Matches any single character.
`%`	Matches any single character that is legal in an Ada identifier.
`$`	Matches Ada delimiters: `& ' ( ) * + , - . / : ; < = > |` When used outside brackets (`[]`), `$` matches beginning and end of line as well.
`\`	Quotes the next wildcard character, causing it to have a literal (not a wildcard) interpretation. `\` must immediately precede the wildcard it quotes.
`{`	Matches the beginning of a line, when used at the beginning of the pattern; otherwise, `{` has a literal meaning.
`}`	Matches the end of a line, when used at the end of the pattern; otherwise, `}` has a literal meaning.
`[]`	Defines a set of characters, of which any one can be matched. The set can be a list (for example, `[ABCDE]`) or a range (for example, `[A-Z]`).
`^`	Excludes the next character or set of characters; `^a` matches any character other than a, and `^[abc]` or `[^abc]` matches any character other than a, b, or c.
`*`	Matches zero or more occurrences of the previous character or set of characters.

ATTRIBUTES

- Specify properties of objects.
- Are postfixed to name components in pathname: `Calculation'Body`
- May or may not accept arguments (see tables, pages QR-9 to QR-12).
  - Arguments are enclosed in parentheses: `My_File'V(5)`
  - Multiple arguments are separated by commas: `My_File'V(4,6)`
- Are often used with wildcards to specify sets of objects by their properties:
  - Multiple attributes denote properties shared by all matched objects: `@'Body'S(Installed)` matches installed unit bodies.
  - Multiple arguments to a single attribute denote disjoint properties: `@'S(Installed,Coded)` matches either installed or coded Ada units.
  - Arguments preceded by ~ denote excluded properties: `@'C(~Binary)` matches all objects except binary files. `@'C(File)'C(~Binary)` matches all files except binary files.

Attributes

Attribute	Description
`'Body`	Resolves to the body of an Ada unit: `"Calculation'Body"`
`'C(`*arguments*`)`	Specifies an object's class or subclass: `"@'C(Library)"`
`'If(`*arguments*`)`	Specifies whether an object is controlled, checked in, checked out, or frozen: `"@'If(Frozen)"`
`'L(`*argument*`)` *or* `'L`	Specifies a library's links for resolution of subsequent name components. Omitting *argument* specifies entire set of links: `"Tools'L.Text_Io"`
`'N(`*nickname*`)`	Specifies an Ada subprogram's nickname: `"Example.Overloaded'N(First)"`
`'S(`*arguments*`)`	Specifies an Ada unit's compilation state: `"@'S(Source)"`
`'Spec`	Resolves to the visible part (specification) of an Ada unit: `"Calculation'Spec"`
`'Spec_View(`*activity*`)` *or* `'Spec_View`	Specifies the spec view listed for a given subsystem in *activity*. Omitting *activity* uses the default activity: `"!Project.Interface.@.Units'Spec_View"`
`'T(`*target_key*`)`	Specifies a library's target key: `"@_Working'T(Mc68020_Bare)"`
`'V(`*arguments*`)`	Specifies an object's version: `"My_File'V(5)"`
`'View(`*activity*`)` *or* `'View`	Specifies the load view listed for a given subsystem in *activity*. Omitting *activity* uses the default activity: `"Command_Interpreter'View"`

Attributes with Predefined Arguments

Arguments for the Conditional Attribute 'If

Short Form	Full Form	Description
Controlled		Matches objects if they are controlled.
In	Checked_In	Matches objects if they are controlled and checked in.
Out	Checked_Out	Matches objects if they are controlled and checked out.
Frozen		Matches objects if they are frozen.

Arguments for the Link Attribute 'L

Argument	Description
Any	Resolves the next name component relative to external and internal links (the default if no argument is specified).
External	Resolves the next name component relative to external links only. External links reference Ada units outside the closest enclosing world.
Internal	Resolves the next name component relative to internal links only. Internal links reference Ada units within the closest enclosing world or any of its subdirectories.

Arguments for the Compilation-State Attribute 'S

Shortest Form	Full Form	Description
A	Archived	Matches units in the archived state.
S	Source	Matches units in the source state.
I	Installed	Matches units in the installed state.
C	Coded	Matches units in the coded state.

Arguments for the Version Attribute 'V

Argument	Description
All	Matches *all* versions of the object.
Any	Matches only the *default* version of the object, which may but need not be the newest version.
Max	Matches the *newest* version of the object, which may but need not be the default version.
Min	Matches the *oldest* retained version of the object.
n	Matches the version with version number *n*. Multiple version numbers can be listed, separated with commas.
−n	Matches the *n*th version preceding the newest version.

Arguments for the Class Attribute 'C: Object Classes

Argument	Description
Ada	Ada program units of any subclass
Archived_Code	Objects appearing in a subsystem view for a code-only unit
File	Files of any subclass
Group	Groups defined for access control
Library	Libraries of any subclass
Null_Device	Devices that accept output and discard it
Pipe	Pipes
Session	User session objects
Tape	Tape drives in the system
Terminal	Terminals in the system
User	Users in the system

Arguments for the Class Attribute 'C: Library Subclasses

Short Form	Full Form	Description
Comb_Ss	Combined_Subsystem	Combined subsystem (can contain only combined views)
Comb_View	Combined_View	Combined view of a subsystem
Directory		Directory
Load_View		Load view of a subsystem
Spec_View		Spec view of subsystem
Subsystem		Spec_Load subsystem (can contain spec, load, or combined views)
System	System_Subsystem	System (object managed using package Cmvc_Hierarchy)
Sys_View	System_View	View in a system
World		World

Arguments for the Class Attribute 'C: Ada Subclasses

Short Form	Full Form	Description
Alt_List	Alternative_List	Insertion point for alternative list
Comp_Unit	Compilation_Unit	Compilation unit that has not been semanticized
Context	Context_List	Insertion point for context clause
Decl_List	Declaration_List	Insertion point for declaration list
Func_Body	Function_Body	Function body
Func_Inst	Function_Instantiation	Generic function instantiation
Func_Ren	Function_Rename	Function rename
Func_Spec	Function_Spec	Function specification
Gen_Func	Generic_Function	Generic function
Gen_Pack	Generic_Package	Generic package
Gen_Param	Generic_Parameter_List	Insertion point for generic parameter
Gen_Proc	Generic_Procedure	Generic procedure
Insertion	Nonterminal	Insertion point
Load_Func	Loaded_Function_Spec	Code-only function
Load_Proc	Loaded_Procedure_Spec	Code-only procedure
Main_Body	Main_Function_Body	Main function body
Main_Body	Main_Procedure_Body	Main procedure body
Main_Func	Main_Function_Spec	Main function specification
Main_Proc	Main_Procedure_Spec	Main procedure specification
Pack_Body	Package_Body	Package body
Pack_Inst	Package_Instantiation	Generic package instantiation
Pack_Ren	Package_Rename	Package rename
Pack_Spec	Package_Spec	Package specification
Pragma	Pragma_List	Insertion point for pragma
Proc_Body	Procedure_Body	Procedure body
Proc_Inst	Procedure_Instantiation	Generic procedure instantiation
Proc_Ren	Procedure_Rename	Procedure rename
Proc_Spec	Procedure_Spec	Procedure specification
Statement	Statement_List	Insertion point for statement
Subp_Body	Subprogram_Body	Subprogram body
Subp_Inst	Subprogram_Instantiation	Generic subprogram instantiation
Subp_Ren	Subprogram_Rename	Subprogram rename
Subp_Spec	Subprogram_Spec	Subprogram specification
Task_Body		Task body

Arguments for the Class Attribute 'C: File Subclasses

Short Form	Full Form	Description
Activity		Activity file (used with subsystem development)
Binary		Binary file
Cmvc_Acc	Cmvc_Access	File containing CMVC access-control information for a view or subsystem
Cmvc_Db	Cmvc_Database	CMVC database (stores source control information in subsystems)
Code_Db	Code_Database	Code saved for a subsystem load view
Compat_Db	Compatibility_Database	Compatibility database for a subsystem
Config	Configuration	Configuration pointer for CMVC
Dictionry	Dictionary	For future development
Documents	Document_Database	Document database (part of Rational Design Facility)
Elements	Element_Cache	Element cache for storing permanent collections of Ada program elements (part of Rational Design Facility)
Exe_Code	Executable_Code	Executable module generated by the linker of the Cross-Development Facility
File_Map	Pure_Element_File_Map	File map
Log		Log file
Mail		Collections of messages (part of Rational Network Mail)
Mail_Db	Mail_Database	User's mailbox (part of Rational Network Mail)
Markup		Text file containing markup, generated from abstract document (part of Rational Design Facility)
Msg_In	Incoming_Mail_Message	For future development
Msg_Out	Outgoing_Mail_Message	For future development
Obj_Code	Object_Code	Relocatable object module generated by the compilation system of the Cross-Development Facility
Objects	Object_Set	Permanent collection of Directory.Object
Ps	Postscript	PostScript file
Search	Search_List	Searchlist file
Switch		Switch file
Swtch_Def	Switch_Definition	Switch definition file
Text		Text file
Venture		A collection of work orders for CMVC
Work	Work_Order	Work order for CMVC
Work_List	Work_Order_List	Work-order list for CMVC

OPTIONS PARAMETER

- Accepts one or more *option specifications*:
  `Options => "options specifications"`
  - Option specifications are strings that assign *values* to *options*.
  - A given option can accept values that are Booleans, predefined literals, or user-specified strings (such as pathnames, time expressions, and so on).

Option Specifications

Syntax	Meaning

option = *value*
or
option => *value*
or
option := *value*
	General format for an option specification; assigns *value* to *option* using any of three delimiters:
	`Options => "After=09/15/92"`
	
	Values containing commas, semicolons, =, =>, or := must be enclosed in parentheses:
	`Options => "Label:=(May 26, 1992)"`
	`Options => "Object_Acl=>(John=>RW)"`

option
or
option = true
	Specifies Boolean option with value True:
	`Options => "Replace"`
	`Options => "Replace => True"`

~option
or
option = false
	Specifies Boolean option with value False:
	`Options => "~Replace"`
	`Options => "Replace := False"`

value
or
option = *value*
	Specifies predefined literal value for *option*:
	`Options => "Fixed_Length"`
	`Options => "Format = Fixed_Length"`

opt = *val*, *opt* = *val*,...
or
opt = *val*; *opt* = *val*;...
	General format for multiple option specifications; uses commas or semicolons as separators (with equivalent meaning):
	`Options =>`
	`"After=09/15/92, Format=R1000"`

opt1 | *opt2* | ... = *val*	Shorthand format for assigning the same value (*val*) to two or more options:
	`Options =>`
	`"Object_Acl|Default_Acl=Retain"`

(~)*option* (~)*option* ...	Shorthand format for specifying multiple Boolean options with either True or False values; uses blanks as separators: `Options => "Replace Promote"`

RESPONSE PARAMETER

- Specifies the response characteristics for a command.
  - Response characteristics include a command's error response, log generation, message output and format, activity, remote-passwords file, and remote-sessions file.
- Accepts special values that specify one of the following prepackaged sets of response characteristics:
  - The *system default profile* (provided by Environment for general use)
  - The *session response profile* (defined in current session-switch settings)
  - The *job response profile* (same as session response profile, unless reset for the job using package Profile commands)
- Accepts special values that filter message output.
- Accepts option specifications that tailor individual response characteristics.

Special Values for Specifying Profiles

Special Value	Description
"<PROFILE>"	Causes the command to obtain its response characteristics from the job response profile.
"<SESSION>"	Causes the command to obtain its response characteristics from the session response profile, ignoring the job response profile.
"<DEFAULT>"	Causes the command to obtain its response characteristics from the system default profile, ignoring the job and session response profiles.

Special Values for Filtering Messages

Special Value	Description
"<ERRORS>"	Logs only negative, error, and exception messages (++*, ***, %%%); perseveres at errors, without raising an exception; otherwise, same as job response profile.
"<IGNORE>"	Logs no messages; perseveres at errors, without raising an exception; otherwise, same as job response profile.
"<NIL>"	Logs no messages; quits at the first error, without raising an exception; uses no activity, remote-passwords, or remote-sessions file; ignores job and session response profiles.
"<PROGRESS>"	Logs only positive, negative, error, and exception messages (+++, ++*, ***, %%%); perseveres at errors, without raising an exception; otherwise, same as job response profile.
"<QUIET>"	Same as "<IGNORE>".
"<RAISE_EXCEPTION>"	Raises an exception at the first error and quits immediately; otherwise, same as job response profile. (For package Archive commands, you should use the string "Raise_Error, <PROFILE>" instead to permit graceful termination after exception is raised.)
"<VERBOSE>"	Logs all messages except debug messages (???); otherwise, same as job response profile.
"<WARN>"	Logs only negative, warning, error, and exception messages (++*, !!!, ***, %%%); perseveres at errors, without raising an exception; otherwise, same as job response profile.

Response Parameter Options

Caution: *Unspecified options are set to nil; to be safe, use options along with special names. Examples:*

```
Response => ("Width=255, <PROFILE>")
Response => ("Use_Error, <PROFILE>")
Response => ("Symbols, <PROFILE>")
```

- `Width = n`

  Sets the width of the message output to the specified number of characters.

- `Reaction = literal`

  Specifies how the command responds to errors:

`Quit`	Stops immediately at the first error; does not raise an exception.
`Propagate`	Stops immediately at the first error; raises an exception.
`Persevere`	Continues processing when an error is encountered; does not raise an exception.
`Raise_Error`	Continues processing when an error is encountered; raises an exception after all processing is complete.

- *`Message-symbols`*

  Boolean options that control whether the corresponding types of messages appear in the log. There are twelve such options, one for each type of message:

  ```
  :::   ???   ---   +++   >>>   ++*
  !!!   ***   %%%   ###   @@@   $$$
  ```

- `Prefix = literals`

  Specifies up to three fields of information to be prefixed to each message:

  ```
  TIME        DATE        SYMBOLS
  HR_MN_SC    MN_DY_YR
  HR_MN       DY_MON_YR
              YR_MN_DY
  ```

- `Log_File = literal`

  Specifies where log messages are to be directed:

`Use_Output`	Directs messages to Current_Output (by default, same as Standard_Output).
`Use_Error`	Directs messages to Current_Error (by default, same as Standard_Error).
`Use_Standard_Output`	Directs messages to Standard_Output (an Environment output window).
`Use_Standard_Error`	Directs messages to Standard_Error (the Environment message window).

- `Activity = activity name`

  Specifies the name of the activity to use during execution.

- `Remote_Passwords = filename`

  Specifies the remote-passwords file to use when accessing remote machines.

- `Remote_Sessions = filename`

  Specifies the remote-sessions file to use when accessing remote machines.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

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