



Fujitsu M2512A : Switch Settings

PCA Layout All configurations marked * are default settings.

Switch SW1

Select SCSI ID

Keys	1	2	3	SCSI ID
	OFF*	OFF*	OFF*	0
	OFF	OFF	ON	1
	OFF	ON	OFF	2
	OFF	ON	ON	3
	ON	OFF	OFF	4
	ON	OFF	ON	5
	ON	ON	OFF	6
	ON	ON	ON	7

SCSI bus parity check

Key 4	ON*	Enabled
	OFF	Disabled

Synchronous data transfer request mode

Key 5	ON	Enabled
	OFF*	Disabled

Changing this function will not affect asynchronous transfers.

Device type

Key 6	ON	Optical memory device (INQ = '07')
	OFF*	Direct access device (INQ = '00')



This determines what response the drive will give to an INQUIRY command.

Spindle automatic stop mode

Key 7	ON*	Enabled
	OFF	Disabled

With the disk loaded, the spindle continues to rotate until the START/STOP unit command instructs the spindle to stop. The spindle automatic stop function comes into operation when the host does not issue a command for about 30 minutes. If the host issues a command after the spindle has stopped then the spindle will start up and process the command without posting a not ready state. ⚡ This function greatly reduces the amount of dust that can collect on the disk, however it is not suitable for a system which requires a quick response as the spindle takes several seconds to get up to speed. ⚡ The access monitoring time and the function itself can be changed or enabled via a MODE SELECT command.

LED mode

Key 8	OFF*	Mode 1
	ON	Mode 2

Mode 1 The LED lights when drive is positioning, reading, writing, loading, ejecting and during the power on test. At power-on test the LED will blink at 1 second intervals after detecting an error or 2 second intervals having detected a thermal error.

Mode 2 The LED lights only on power-on test and whilst waiting. Error detection and reporting is as for Mode 1.

Switch Bank SW2

Write cache mode.

Key 1	ON	Enabled
	OFF*	Disabled

When enabled, a write error is reported at the completion of the **next** command. This can speed up write times but do ensure your system will respond correctly by knowing what data to resend.

Key 2	OFF*	Not Used
--------------	------	----------

Mac mode

Key 3	ON	Enabled
	OFF*	Disabled

Because of the differing ways that both Apple Macintosh and PC hosts respond to the status of the drive, set this switch accordingly. Mac mode ON will disable the 'UNIT ATTENTION' reporting on



power up and media change.

Verify after write mode

Key 4	ON	Disabled
	OFF*	Enabled

When disabled, the performance of a write operation is improved by approximately 20%, however data integrity cannot be guaranteed.

SCSI level

Key 5	ON	SCSI-2 mode
	OFF*	SCSI-1 mode (Compatible with M2511A)

SAVE DATA POINTER message mode

Key 6	OFF*	SAVE DATA POINTER message posted before sending DISCONN
	ON	No message sent before DISCONNECT message

Key 7	OFF*	Not used
--------------	------	----------

Key 8	OFF*	Factory examination mode
--------------	------	--------------------------

Terminal CNH 1

SCSI Terminating Power

1-2	3-4	Function
SHORT*	SHORT*	Power is supplied from the drive to both the terminatin the TERMPWR pin on the SCSI bus. (Pin 26)
OPEN	SHORT	The TERMPWR pin is not used. Power is supplied from the active terminator.
SHORT	OPEN	Power is not supplied to the terminating resistor from the TERMPWR pin.
OPEN	OPEN	Not used

SCSI Active Terminator Module on Drive PCA

5-6	SHORT*	Enabled
	OPEN	Disabled



Note that there is no removable terminating resistor module on this drive. All termination functions are set through this terminal.

Terminal CNH 2

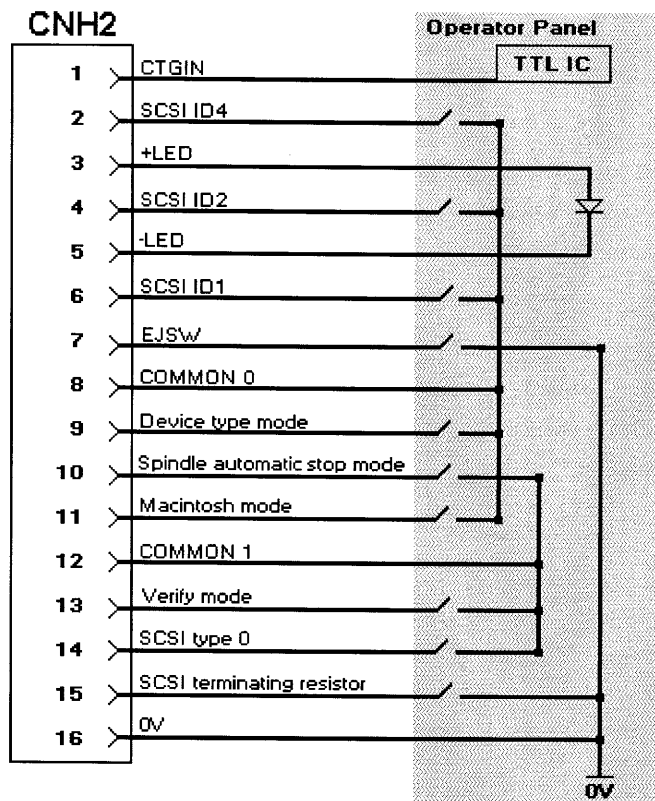
External operator panel connector

Pin	Signal
1	CTGIN (Cartridge in)
2	-ID2 (bit value 4)
3	+LED (+5V)
4	-ID1 (bit value 2)
5	-LED
6	-ID0 (bit value 1)
7	EJSW (Eject switch)
8	COMMON 0
9	Device type mode
10	Spindle automatic stop mode
11	Macintosh mode
12	COMMON 1
13	Verify Mode
14	SCSI type 0
15	SCSI terminating resistor
16	0V (Ground)




The cable length to the operator panel should not exceed 30 centimetres

COMMON 0 and COMMON 1 are control signals and should not run to ground.






[products](#) [support](#) [partners](#) [news+info](#) [jobs](#) [where to buy](#) [buy online](#)

[DISC](#) [TAPE](#) [SOFTWARE](#) [CONTACT US](#)

[support](#) [SITE INDEX](#) [GO](#) [Shortcuts](#)

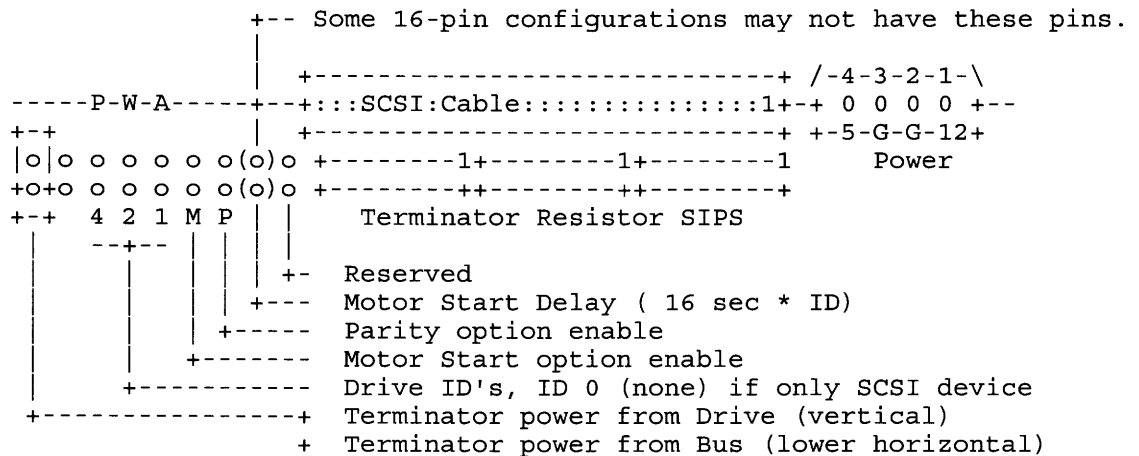
[SEARCH](#)

Specifications for ST-41200N

This text file can be obtained by ftp. ASCII linedraw characters are used that display normally in textmode. Windows based environments can use the Letter Gothic Line TrueType font to display and print ASCII linedraw characters.

SEAGATE TECHNOLOGY, INC.
 1-800-SEAGATE
<http://www.seagate.com>
 (C)opyright 1991

ST-41200N (94601-1200) Wren 7 SCSI and SCSI-2



ST-41200N
 94601-12G/M WREN 7 FH

UNFORMATTED CAPACITY (MB)	_____	1.2G
FORMATTED CAPACITY (xx SECTORS) (MB)	_____	1037
AVERAGE SECTORS PER TRACK	_____	71
ACTUATOR TYPE	_____	VOICE COIL
TRACKS	_____	28965
CYLINDERS	_____	1931
HEADS	_____	15
DISCS	_____	8
MEDIA TYPE	_____	THIN FILM
RECORDING METHOD	_____	ZBR RLL (1,7)
TRANSFER RATE (mbytes/sec)	_____	1.875-2.875
SPINDLE SPEED (RPM)	_____	3,600
AVERAGE LATENCY (mSEC)	_____	8.33
BUFFER	_____	256 Kbyte
SCSI-1: Read Look-Ahead, Non_Adaptive, Single-Segmented Buffer		
SCSI-2: Read Look-Ahead, Adaptive, Multi-Segmented Cache		
INTERFACE	_____	SCSI-2
TPI (TRACKS PER INCH)	_____	1600
BPI (BITS PER INCH)	_____	32750



AVERAGE ACCESS (ms)	_____	15
SINGLE TRACK SEEK (ms)	_____	2.5
MAX FULL SEEK (ms)	_____	34
MTBF (power-on hours)	_____	150,000
POWER REQUIREMENTS: +12V START-UP (amps)	_____	4.5
+12V TYPICAL (amps)	_____	1.6
+5V START-UP (amps)	_____	1.1
+5V TYPICAL (amps)	_____	0.8
TYPICAL (watts)	_____	24
MAXIMUM (watts)	_____	60
BUFFERED STEP PULSE RATE (micro sec)	_____	
WRITE PRECOMP (cyl)	_____	N/A
REDUCED WRITE CURRENT (cyl)	_____	N/A
LANDING ZONE (cyl)	_____	AUTO PARK
IBM AT DRIVE TYPE	_____	0 or NONE

ZBR = Zone Bit Recording = Variable sectors per track
Already low-level formatted at the factory.

Seagate reserves the right to change, without notice, product offerings or specifications. (6/26/90)

[Products](#) | [Support](#) | [Partners](#) | [News + Info](#) | [Jobs](#) | [Where to Buy](#) | [Buy Online](#) | [Contact Us](#) | [Site Index](#)

Copyright © 2001, Seagate Technology | [Privacy Policy](#) | [Legal](#)



Seagate products support partners news+info jobs where to buy buy online

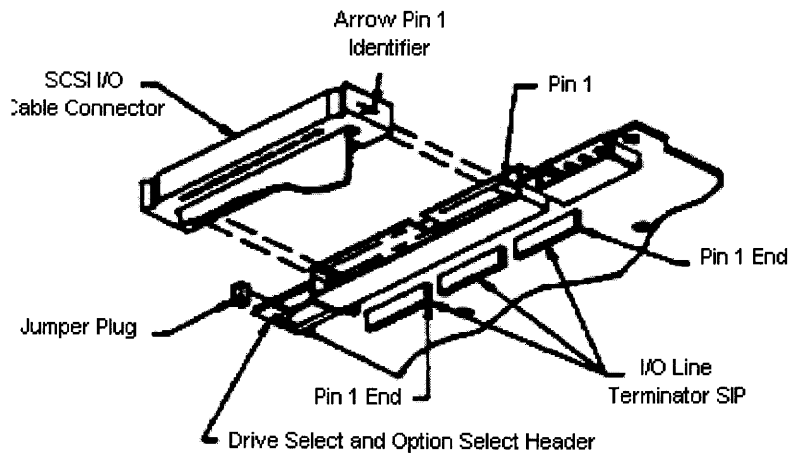
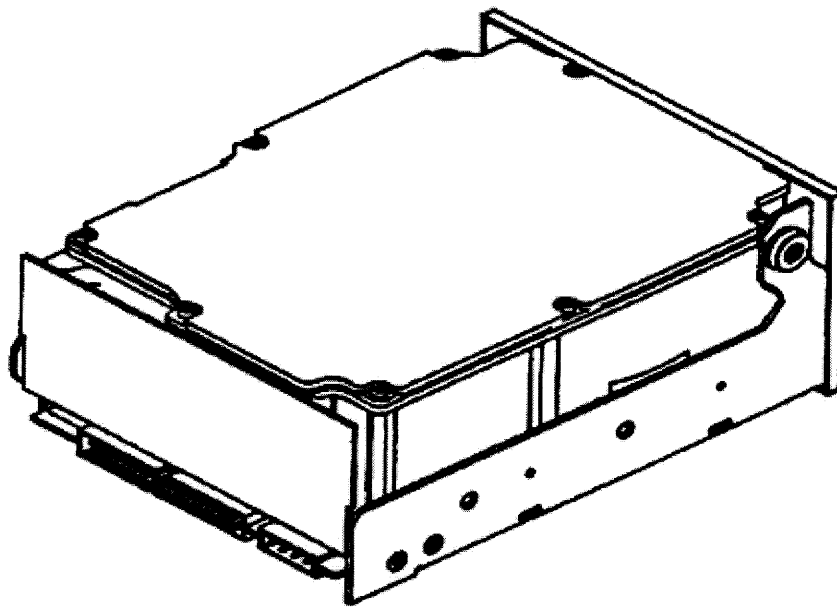
DISC TAPE SOFTWARE CONTACT US

disc SITE INDEX SEARCH Shortcuts

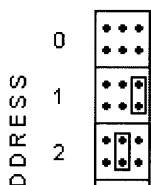
ST41200N

Wren 7

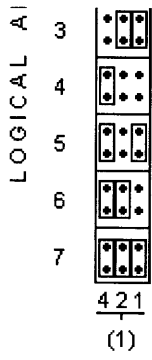
Capacity: 1.04 GB
 Speed: 3600 rpm
 Seek time: 15 ms
 SeaFAX#: 412006



Drive Select Jumpers







[Products](#) | [Support](#) | [Partners](#) | [News + Info](#) | [Jobs](#) | [Where to Buy](#) | [Buy Online](#) | [Contact Us](#) | [Site Index](#)

Copyright © 2001, Seagate Technology | [Privacy Policy](#) | [Legal](#)



Guide to Machine Initialization

Guide to Machine Initialization

Release D_12_6_5



Copyright © 1991 by Rational

Part Number: 508-003207-006

May 1992 (Software Release D_12_6_5)

PostScript is a registered trademark of Adobe Systems Incorporated.

Rational and R1000 are registered trademarks and Rational Environment is a trademark of Rational.

Sun Workstation is a registered trademark of Sun Microsystems, Inc.

UNIX is a registered trademark of AT&T.

Rational
3320 Scott Boulevard
Santa Clara, California 95054-3197



Guide to Machine Initialization



Contents

1. Overview	1
1.1. World !Machine.Initialization.Rational	2
1.2. Worlds !Machine.Initialization.[Site,Local]	2
2. Setting Up the Site and Local Worlds	4
3. Hints for Implementing System Customizations	5
3.1. Writing Customized Initialization Procedures	5
3.2. Using “_Start” Files to Reference Initialization Procedures	6
3.3. Controlling the Order of Execution	7
3.4. Customizing Disk-Collection Thresholds	8
4. Enabling and Configuring Login Ports	8
4.1. Enabling Ports for Login	8
4.2. Customizing Port Characteristics	10
4.3. A Sample Terminal_Configuration File	11
4.4. Terminal-Configuration Options	11
5. Configuring Printers	13
5.1. Where to Specify Printer Information	14
5.2. Adding Entries to a Printer_Configuration File	15
5.3. Specifying a Directly Connected Printer	16
5.4. Specifying a Networked Printer	17
5.5. Specifying an Environment File	17
5.6. Specifying a Workstation Directory	18
5.7. Associating Default Printers with Individual Users	20



D_12_5_0 changed the software that is automatically executed by the Environment during the boot process. These changes apply to D_12_5_0 and all subsequent releases (including D_12_6_5). In particular, the software was reorganized to make it easier to:

- Install layered products
- Distinguish Rational-specific, site-specific, and machine-specific customizations
- Configure a network of printers for large sites

The following subsections describe the D_12_5_0 mechanisms for initializing an R1000. Change bars indicate sections that have been added to this document since D_12_5_0; the initialization mechanisms themselves have not changed. You can read this document online by traversing to `!Machine.Initialization.Guide_To_Machine_Initialization` or by executing `What.Does` (`"!Machine.Initialization"`).

See also the "Installation Procedure" for specific steps to convert any existing initialization software to the D_12_5_0 mechanisms.

1. 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 prior 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 world `!Machine.Initialization`, 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 world `Rational` 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 worlds `Site` and `Local` 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 world `Site` is intended for customer-written objects that are common to two or more machines at a given site.



- The world Local is intended for customer-written objects that apply only to the current R1000.

The following subsections give more detail about these worlds.

1.1. World !Machine.Initialization.Rational

The Rational world contains objects supplied by Rational 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 CDFs, RDF, Rational 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, disk-collection thresholds, and so on
- 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.

1.2. Worlds !Machine.Initialization.[Site,Local]

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 reference 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 section 3.2.)
- *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 sections 4 and 5.)
- 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 world !Machine.Initialization.Rational.)

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 console. (The standard is to route only warning, problem, and fatal messages to the operator 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 section 3.4). Note, however, that values set by the procedure in world Rational 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



world Rational should be sufficient; see your Rational technical representative if you want to use different settings.

2. 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 section 1.2. Create the appropriate procedures and/or “_Start” files, using the hints given in section 3.
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 section 4.
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 section 5.
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 world !Machine.Initialization.Rational. Create any files required for initializing these products (for example, 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.”

After your R1000 has been upgraded to D_12_5_0 or a more recent release, you can use the following guidelines to set up your Site and Local worlds:

1. 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 section 3 for information about initialization procedures and “_Start” files.
2. Inspect the default Terminal_Configuration file that was created in the Local world during installation. Edit this file if you want to enable a different set of ports and/or specify further connection or communication information; see section 4.
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 section 5.



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 world `!Machine.Initialization.Rational`. Create any files required for initializing these products (for example, a text file that registers an RDF customization).

3. Hints for Implementing System Customizations

This section provides information about:

- Writing customized initialization procedures in the Site and Local worlds; see section 3.1.
- Writing “_Start” files that reference procedures that reside in other Environment libraries; see section 3.2.
- Controlling the order in which the customized initialization procedures and the “_Start” files are processed by the Start procedure; see section 3.3.
- Customizing disk-collection thresholds; see section 3.4.

3.1. 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 sections 4 and 5).
- 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 section 3.4).

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 section 3.3).



3.2. 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 section 3.3).

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. Note that 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 section 3.3). 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 section 3.3.

Using a “_Start” file is equivalent to executing the referenced procedure via `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.

3.3. 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 `Program.Run`.

- 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 `Program.Run_Job`.

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.



3.4. 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 world `Rational` are calculated based on your disk configuration and should be sufficient; see your `Rational` technical representative if you want to use different thresholds.

4. 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 section 4.1). 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 section 4.4.

4.1. 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 section 4.4.)



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 containing 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 summarized in Table 1.

Table 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.



4.2. 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 section 4.4.

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 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 2 and 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.

4.3. 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)
```

4.4. Terminal-Configuration Options

Tables 2, 3, and 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 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 3 Enumeration Option for Specifying Terminal Type

Option => Value	Description
Terminal_Type	Specifies the output driver type for a given port. <i>Value</i> can be any valid terminal type name, including (but not limited to): Cit500r Facit Rational Vt100 Xrterm

Table 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. <i>Value</i> 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. <i>Value</i> can be: None Xon_Xoff Dtr Rts



Table 4 Enumeration Options for RS232 Communication Characteristics (continued)

Option => Value	Default Value	Description
Input_Rate	Baud_9600	Sets the incoming data rate for a given port. <i>Value</i> 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. <i>Value</i> can be: None Odd Even
Stop_Bits	2	Sets the number of stop bits for a given port. <i>Value</i> 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. <i>Value</i> can be: None Xon_Xoff Dtr Rts
Receive_Xon_Xoff_Bytes	(17,19)	Specifies flow-control bytes so that the R1000 can regulate the data it receives on the specified port. <i>Value</i> is (<i>n</i> , <i>m</i>), where <i>n</i> and <i>m</i> 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. <i>Value</i> is: (<i>n</i> , <i>m</i>), where <i>n</i> and <i>m</i> are natural numbers in the range 0..255.

5. 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 also 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 file-driven mechanism, you do not need to use procedures from package Queue (such as Add, Create, Enable, and Register) to do these things.



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 either through commands in package Queue or through 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.

5.1. 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 doesn't 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.



5.2. 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`.)

5.3. 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 section 5.2. 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)
```



5.4. 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 section 5.2. 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)
```

5.5. 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 section 5.2. 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)
```

5.6. 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 section 5.2. 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 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
```



Guide to Machine Initialization

```
#
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
```

5.7. 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.



Erlo Haugen

From: Greg Bek [gab@rational.com]
Sent: 20. april 2002 00:13
To: Erlo Haugen
Subject: RE: TCP/IP routing

Try this:

Best I can come up on this (from various docs and notes)

Route information would go into file
!machine.transport_routes

Router table would be loaded with
!commands.Transport_Route.Load

Route information can be shown with command
!Commands.Transport_Route.Show

To define a default IP route:

- 1) Add a line to !Machine.Transport_Routes that designates an IP router, with no other text and no whitespace on the line. The router can be designated by:
 - Its IP address (in decimal-dotted notation)
 - or
 - A name that can be resolved to an IP address via !Machine.Transport_Name_Map.You must use a name that is defined in !Machine.Transport_Name_Map,

because it is not possible to communicate with the name server (the machine designated in !Machine.Tcp_Ip_Name_Server) at the time the name is resolved.

- 2) After editing (and closing) !Machine.Transport_Routes, execute Tcp_Ip_Boot to load the default IP route into the Ethernet controller.

To change the default IP route:

- 1) Edit !Machine.Transport_Routes to designate the new default IP router in a line with no whitespace and no other information (see above).
- 2) Execute Transport_Route.Undefine to erase any default IP routes that are presently defined. You can find them by running Transport_Route.Show.
- 3) After editing (and closing) !Machine.Trasport_Routes, execute Tcp_Ip_Boot to load the default IP route into the Ethernet controller.

You can define other, nondefault IP routes using other IP routers, as described in package Transport_Route, but any nondefault route that is not used for a period of 30 minutes is likely to be deleted by the Series 400 controller, and subsequent datagrams to which the route pertained will be transmitted to the default IP router (until an ICMP Redirect message is received).

If you must use a nondefault route in a 400, and the default IP router cannot redirect you to it periodically, you can define it in !Machine.Transport_Routes and then execute a keep alive program to prevent it from being deleted due to disuse. An example of a keep alive routine is:

```
Scheduler.Set_Job_Attribute (System_Uilities.Get_Job,
```



```
"Kind","Server");
loop
  Transport_Route.Load;
  delay 25 * 60.0;
end loop;
```

Transport_Route.Show will display a route that is deleted due to disuse as though it were still in effect. The only way to discover the contents of the 400 IP routing table is to transmit test IP datagrams and observe (on the Ethernet) the IP routers to which they are relayed.

Greg

> -----Original Message-----

> From: Erlo Haugen [mailto:elh@terma.com]

> Sent: Friday, April 19, 2002 6:25 PM

> To: 'Greg Bek'

> Subject: RE: TCP/IP routing

>

>

> I have checked on our machines. None of them has the mentioned file.

> Will a restart of the machine be necessary if I create this file ?

>

> Have a nice weekend!

>

> Erlo Haugen

> Systems Software Engineer

>

> Terma A/S

> Mårkærvej 2

> DK-2630 Tåstrup

>

> +45 43 52 15 13

> elh@terma.com

> www.terma.com

>

>

>

> -----Original Message-----

> From: Greg Bek [SMTP:gab@rational.com]

> Sent: 19. april 2002 05:53

> To: Erlo Haugen

> Subject: RE: TCP/IP routing

>

> I haven't heard back yet.

>

> >From memory it is a file under !Tools.Networking

>

> > It could be the file !Tools.Networking.Default_Route

> > and you just put the routers IP address in there. But

> > I'm not sure.

>

> > But's it's been 6+ years since I last set up an R1000 so

> > the pure memory is a big foggy. If I was sitting logged

> > into one I'd get the answer quickly.

>

> > G

>



Rational Environment Release Information

Release D_12_7_3



Copyright © 1992 by Rational

Part Number: 508-003207-007

November 1992 (Software Release D_12_7_3)

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

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

PostScript is a registered trademark of Adobe Systems Incorporated.

Rational and R1000 are registered trademarks and Rational Environment is a trademark of Rational.

Sun Workstation is a registered trademark and OpenLook, NeWS, and SunOS are trademarks of Sun Microsystems, Inc.

UNIX is a registered trademark of UNIX System Laboratories, Inc.

VMS is a trademark of Digital Equipment Corporation.

X Window System is a trademark of MIT.

Rational
3320 Scott Boulevard
Santa Clara, California 95054-3197



1. Overview

The D_12_7_3 release of the Rational Environment™ is primarily a maintenance release that:

- Supports installation of Environment releases over a network (section 6.1).
- Simplifies the process of crash analysis and recovery (sections 6.2, 6.3, 7.6).
- Introduces a new release of the Rational X Interface (RXI) (section 6.5).
- Introduces Rational Access, an OSF/Motif™-based graphical user interface for the Environment. Access runs as an X Window System™ application on a UNIX® workstation. It features:
 - The OSF/Motif mouse, menu, and dialog box paradigm.
 - Many new “full service” commands, which combine several lower-functionality Environment operations.
 - A comprehensive, menu-based online help facility.
 - Platform-independent key and mouse bindings.
 - Full support for existing Environment features such as “item-operation” combinations and command windows.

For complete information, see the *Rational Access User's Guide* and *Rational Access Release Information* delivered with this release.

- Fixes a large number of problems from previous releases, enabling existing commands to work correctly. Some of these fixes introduce new features (see section 6); other fixes involve changed features (see section 7).
- Introduces a new, up-to-date *System Manager's Guide*, containing combined information for Series 200, 300, and 400 system (see section 8.1).
- Introduces new online help for each of the packages described in the Debugging book (Volume 3) and the Project Management book (Volume 11) of the *Rational Environment Reference Manual* (see section 8.2).

Note: This release information can be found online in the !Machine.Release.Release_Notes world in line-printer (Environment_Release12_7_3_Lpt) and PostScript® (Environment_Release12_7_3_Ps) formats.

2. Supported Configurations and Upgrades

D_12_7_3 supports the following configurations of the R1000®:

- Series 200 (Models 10, 20, and 40)
- Series 300 System (300S)
- Series 300 Coprocessor (300C) for the IBM® RISC System/6000™ and Sun Workstation® servers
- Series 400 System (400S)
- Series 400 Coprocessor (400C) for the IBM RISC System/6000 and Sun Workstation servers



D_12_7_3 supports two kinds of tape drive:

- The 9-track tape drive, which is standard on the Series 100 and 200, and an optional upgrade to the Series 300S
- The 8-millimeter tape drive, which is standard on the Series 300S, 300C, 400S, and 400C, and an optional upgrade to the Series 200

D_12_7_3 also supports the optional expansion of memory from 32 megabytes to 64 megabytes to improve system performance. This upgrade applies to all series except the Series 100. The combinations of configurations and upgrades supported by D_12_7_3 are shown in Table 1.

Table 1 Configurations and Upgrades Supported by D_12_7_3

Configuration	8-mm Tape Drive	9-Track Tape Drive	32-Mb Memory	64-Mb Memory
Series 200	Upgrade	Standard	Standard	Upgrade
Series 300S	Standard	Upgrade	Standard	Upgrade
Series 300C	Standard	N/A	Standard	Upgrade
Series 400S	Standard	N/A	Standard	Upgrade
Series 400C	Standard	N/A	Standard	Upgrade

3. Compatibility

D_12_7_3 is fully compatible with all production versions of Rational layered-software products (Table 2).

Table 2 Compatibility with D_12_7_3

Rational Product	Compatible Release
CDF: Mc68020_Bare	5_1_2 or later
CDF: Mc68020_Os2000	6_1_3 or later
CDF: Mc68020_Hp_Unix	6_2_4 or later
Design Facility: 2167	6_0_7 or later*
Design Facility: 2167A	6_2_5 or later*
Design Tools	10_2_9 or later
Documentation Tools	10_2_9 or later
Insight	1_3_0
Mail	11_4_5 or later
Rational Access	1_0_1
Rational Publishing Interface	1_0_2 or later



Table 2 Compatibility with D_12_7_3 (continued)

Rational Product	Compatible Release
Rational Teamwork Interface	2_1_2 or later
Rational X Interface	10_5_2 or later
Rational Windows Interface	10_1_1 or later
Remote Compilation Facility	1_1_1 or later
RCF: Ibm_Rs6000_Aix	1_1_0 or later
RPC	1_0_2 or later
Software Analysis Workstation	6_0_0 or later
Target Build Utility	10_0_3 or later
TestMate	2_0

* As delivered, D_12_7_3 is compatible only with Design Facility:2167A release 7_2_1. However, other releases and customizations can be made compatible by recompiling them with the new LRM Interface (see section 7.11).

4. Upgrade Impact

The Environment can be upgraded from to D_12_7_3 without forcing you to Archive.Save and Archive.Restore your applications. You will not have to modify or recompile any of your own tools, with the possible exception of:

- Tools, Rational Design Facility releases, or Rational Design Facility customizations written against the LRM Interface. In particular, changes made to the Declaration_Kinds, Attribute_Designator_Kinds, or Pragma_Kinds enumerations may result in compilation or runtime errors. See section 7.11 for details.

The procedure Find_Lrm_Change_Candidates, located in the Pdl_Commands subsystem, can be used to assist in identifying potential compatibility problems that are not caught by the compiler.

- Tools written against the unit specifications listed in "Impact of Specification Changes," below.
- Customizations of unit bodies listed in "Impact of Implementation Changes," below.

The new declarations listed in section 6 are all installed upward-compatibly and therefore have no impact on user-written tools.

Rational does not provide tools for reverting to previous releases of the Environment. You will be able to revert to the previous release, however, if you make a complete backup of the Environment, including the Diagnostic File System (DFS), before you upgrade.



4.1. Impact of Specification Changes

The installation process for D_12_7_3 changes several Environment specifications. The installation process attempts to make all changes incrementally; however, if incremental changes fail, the installation process overwrites the necessary specifications. Overwriting these specifications will cause the demotion of any customer-created tools written against them. The installation process tries to recompile such tools automatically; however, depending on the nature of the tools, some may require modification before they can be recompiled. Units that cannot be recompiled during installation are listed in the installation log.

The unit specifications that are overwritten depend on the release from which you are upgrading.

4.1.1. Upgrading from D_12_6_5 or Later

The following unit specifications are changed when you upgrade from D_12_6_5, D_12_6_6, or D_12_6_7 to D_12_7_3:

- !Implementation.Activity_Implementation'Spec
- !Commands.Cmvc'Spec

4.1.2. Upgrading from D_12_5_0

In addition to the unit specifications overwritten when upgrading from D_12_6_5 (or later), upgrading from D_12_5_0 overwrites the following unit specifications:

- !Commands.Library'Spec
- !Commands.System_Backup'Spec
- !Tools.Access_List_Tools'Spec
- !Tools.Code_Segment_Object_Editor'Spec
- !Tools.Library_Object_Editor'Spec

If you are upgrading from D_12_5_0, you should also read the release information for D_12_6_5; release information is located online in !Machine.Release.Release_Notes.

4.1.3. Upgrading from D_12_2_4

In addition to the unit specifications overwritten when upgrading from D_12_6_5 (or later) and D_12_5_0, upgrading from D_12_2_4 overwrites the following unit specifications:

- !Commands.Abbreviations.Print'Spec
- !Commands.Access_List'Spec
- !Commands.Archive'Spec
- !Commands.Queue'Spec
- !Commands.Remote_Passwords'Spec
- !Tools.Networking.Tcp_Ip_Boot'Spec



- !Tools.Networking.Transport'Spec
- !Tools.Networking.Transport_Defs'Spec
- !Tools.Networking.Transport_Name'Spec

Furthermore, the units in !Machine.Initialize@ are either demoted or moved to other locations to accommodate the D_12_5_0 mechanisms for initializing an R1000; for complete details, see Appendix E, "Machine Initialization," in the *System Manager's Guide* delivered with this release or the online information in !Machine.Initialization.Guide_To_Machine_Initialization.

If you are upgrading from D_12_2_4, you should also read the release information for D_12_5_0 and D_12_6_5; release information is located online in !Machine.Release.Release_Notes.

4.1.4. Upgrading from D_12_1_1

In addition to the unit specifications overwritten when upgrading from D_12_6_5 (or later), D_12_5_0, and D_12_2_4, upgrading from D_12_1_1 overwrites the following unit specification:

- !Implementation.Work_Order_Implementation'Spec

If you are upgrading from D_12_1_1, you should also read the release information for D_12_2_4, D_12_5_0, and D_12_6_5; release information is located online in !Machine-Release.Release_Notes.

4.2. Impact of Implementation Changes

Because unit bodies may contain user customizations, an overwritten unit body's contents are saved in a text file in the same library as the overwritten body. The name of the file is of the form *Unit_Name_Vnn*, where *nn* is the unit's default version number. These customizations then can be transferred to the new implementation.

The unit bodies that are overwritten by the D_12_7_3 installation depends on the release from which you are upgrading.

4.2.1. Upgrading from D_12_6_5 or Later

No unit bodies are overwritten when you upgrade from D_12_6_5, D_12_6_6, or D_12_6_7 to D_12_7_3.

4.2.2. Upgrading from D_12_5_0, D_12_2_4, or D_12_1_1

Upgrading from D_12_5_0, D_12_2_4, or D_12_1_1 overwrites the following unit body:

- !Commands.Abbreviations.Print'Body

5. Known Problems

There are no known problems introduced by this D_12_7_3 release.



6. New Environment Features

D_12_7_3 provides new interfaces and features, including:

- New ability to install Environment releases over the network (see section 6.1).
- New ability to specify who, if anyone, the system is to notify after a crash (section 6.2).
- New procedures in package Dfs for setting Failure Reboot options (section 6.3).
- New procedures in the System_Maintenance subsystem for displaying configuration information and displaying the session associated with a job (section 6.4).
- A new release of the Rational X Interface (RXI) (section 6.5).
- New world containing additional Network_Public sessions to be used by the Archive Server to alleviate token problems (section 6.6).

6.1. Network Installation

The D_12_7_3 release is distributed with a set of network installation tools. This feature is only of interest to those sites that have multiple machines connected via a network, and only at the time the D_12_7_3 release is installed.

In previous releases, the installation tapes for a new Environment release had to be separately loaded on each machine that was to be upgraded. With the D_12_7_3 release, the installation tapes only have to be loaded on one machine on the network. All of the other machines can be upgraded over the network.

Refer to the *D_12_7_3 Installation Procedure* for details.

6.2. Crash Notification

Since the introduction of tombstone files in the D_12_5_0 release, the system has produced an analysis of the previous outage; this analysis is written to the system error log. In D_12_7_3, system managers can specify additional distribution of the analysis output in a new control file. The pathname of this file is:

```
!Machine.Initialiation.Local.Previous_Outage_Configuration
```

This file contains information about who to notify by mail after the system reboots and where to put or send information about the failure. For more information about this file and other changes made to the process of crash analysis and recovery, see section 7.6 or Appendix B, "Diagnostic Crash Procedures for Release D_12_7_3" in the *System Manager's Guide* delivered with this release.

6.3. New Procedures in Package Dfs

In D_12_7_3, the system can be configured to reboot following a failure as long as no hard failure has been detected. This feature is known as *Failure Reboot*.

New procedures in package !Machine.Dfs'Spec_View.Units.Dfs control the Failure Reboot feature. They are described in the following subsections.



For more information about the new Failure Reboot feature, see section 7.6 or Appendix B, "Diagnostic Crash Procedures for Release D_12_7_3" in the *System Manager's Guide* distributed with this release.

6.3.1. Reboot_On_Failure

```
procedure Reboot_On_Failure (Enabled : Boolean := True)
```

Sets whether the R1000 will automatically reboot after a failure. If True, the system attempts to automatically reboot after a system failure.

6.3.2. Reboot_On_Failure_Interval

```
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.

6.3.3. Quiesce_Reboot_On_Failure

```
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.

6.3.4. Reboot_On_Failure_Settings

```
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
```

6.4. New Procedures in System_Maintenance

D_12_7_3 includes two new procedures in the !Commands.System_Maintenance subsystem.

6.4.1. Show_Elaborated_Configuration

```
procedure Show_Elaborated_Configuration;
```

Displays the configuration currently elaborated on this R1000. The configuration is the versions on the operating system subsystems that make up an Environment release. This procedure



displays the name of the current configuration, the subsystems and their versions that the configuration contains, and the microcode version that is running.

This information may be helpful to Rational personnel when they are investigating a problem on the machine. Running this command will not cause any harm to the R1000, but the output will have very little meaning to the end users.

6.4.2. Show_Session_Of_Job

```
procedure Show_Session_Of_Job (Job : Machine.Job_Id := Default.Process);
```

Displays the Session Id of the specified job. If no value is specified for the Job parameter, then the job that is calling this procedure is used. This procedure displays the job number and name, the user identity that the job is running under, the base session name that the job is running under, and the full session name that the job is running under. This procedure is useful in determining who started a given job, or under what identity a given job is executing.

6.5. Rational X Interface Enhancements

D_12_7_3 includes a new release (10_10_1) of the Rational X Interface (RXI). This release includes the following enhancements:

- A terminal type and key binding procedures have been added for XSun101 and XNews101 to support the Sun 101 keyboard. If you use these terminal types, you should use the IBM RS/6000 RXI supplement and keyboard overlays.
- An upgrade to SunOS™ 4.1.2, VMS™ 5.4, Sun NeWS™/OpenLook™ 3.0, and MIT X Window System X11R5.
- A new directory containing PostScript copies of all RXI keyboard overlays. This directory is !Machine.Editor_Data.Keyboard_Overlays. To print a keyboard overlay from the Environment, specify the Original_Raw option to the Print command.

The basic behavior and features of RXI have not been changed.

6.6. New World Network_Public_Archive_Server_Sessions

The new world !Machine.Network_Public_Archive_Server_Sessions contains objects for seven new sessions to be used by the Archive Server. These sessions are called Network_Public_Session_1 through Network_Public_Session_7. For more information about these sessions and changes to the Archive Server, see section 7.3.6.

7. Changes from D_12_6_5

This section describes the changes, enhancements, and user-visible problem fixes that D_12_7_3 makes to existing features of the Environment. The information in this section is presented in roughly the same order in which it would be found in the *Rational Environment Reference Manual* (ERM).

- Basic editing operations such as selection and definition (EI and EST) (section 7.1)



- Session and job management (SJM)—specifically, package What (section 7.2)
- Library-management (LM), including changes to archive (section 7.3), compilation (7.4), and other library-management facilities such as access control (7.5)
- System-management (SMU), including changes to crash analysis (section 7.6, printing, and messages (7.7)
- Subsystems, CMVC, and activities (PM) (sections 7.8 and 7.9)
- Networking facilities such as Telnet and FTP (section 7.10)
- LRM Interface (section 7.11)

7.1. Changes to Package Common

7.1.1. Selection in Environment Menus

Selection of entire Environment menus has been fixed. In previous releases, selecting something in a menu would fail if:

- You had two menus displayed on your screen, and
- One menu was selected completely, and
- You attempted to select anything in the other menu.

For example, if an obsolescence menu and an xref menu were displayed and all of the entries in the obsolescence menu were selected, you could not select anything in the xref menu. In D_12_7_3, the new selection occurs and the other menu is deselected. PRS numbers 2115837-Etoi-Phl and 8995225-Shei-Jst.

7.1.2. Locks and Program_Errors

The Environment no longer generates a Program_Error message in the following two cases:

- If an object is locked and being written to by another job or user and you attempt to execute a search command in the image of that object. PRS number 5098655-Mago-Sdj.
- If you execute Common.Definition on a large text file and attempt to perform an operation in that image before it is completely displayed. PRS number 8957642-Cook-Swb.

In both of the above cases the Environment now generates an appropriate lock message.

7.2. Changes to Package What

When What.Locks is applied to a deleted unit, it no longer fails with an undecipherable message. The new error message correctly identifies the problem. PRS numbers 10464-Star and 5476876-Voya-Phil.

When What.Object is executed on a window that does not correspond to a library object (such as an I/O window or a mail message), it now fails with an appropriate message: **What.Object failed - <IMAGE> does not refer to a directory object.** PRS number 12728-Star.



What.Object no longer fails when:

- It is executed on a window that contains the image of an associated file (indicated in the library display by pointy brackets). PRS number 167263-Clem-Marl.
- It is executed in a text file and the cursor is located inside a selection within that file. PRS number 4317752-Clem-Marl.

The output from the What.Users command now contains two or more entries for the identity Network_Public. One entry contains all of the jobs that were traditionally run under the session !Machine.Network_Public_Session, such as the Print Queue Server and the Mail Distribute Server. The additional entries each contain only one job, an Archive Sever job. These additional Network_Public entries are the result of changes to the Archive Server (see section 7.3.6). In order to determine the session under which a particular job (such an Archive Server job) is running, see the new Show_Session_Of_Job command (section 6.4).

7.3. Changes to Archive

7.3.1. Archiving into Subsystems

The Archive.Copy and Archive.Restore commands no longer allow you to create:

- A subsystem within an existing subsystem
- A world within a subsystem view

If an attempt is made to Copy or Restore a subsystem within a subsystem, the command is terminated with the error **Bad Subsystem Structure**. Note that when copying a subsystem from one R1000 to another across the network, if the restore part on the target fails with this bad subsystem structure error, then the restore part of the archive terminates — however, the sending part of the Archive operation continues sending units until it realizes that the receiver has terminated. At that point, the sender terminates with a **connection broken** error. You must look earlier in the log to see the “real” error message indicating that you are not allowed to create subsystems within subsystems.

If an attempt is made to Copy or Restore a world into a subsystem view, the command now converts the world into a directory before copying or restoring it. PRS number 0-0298-0.

7.3.2. Archiving Loaded Main Programs

In previous releases, the Archive.Copy command failed if you attempted to copy a loaded main program (Load_Proc or Load_Func) to another location or another machine where:

- A coded older version of that loaded main program already existed, and
- The archive job did not have full access to replace the existing loaded main program. but had enough access to demote the existing loaded main program

In such a case, the copy would fail but the existing version of the loaded main program would be left in archived state and in some cases could not be promoted back to coded state.



In D_12_7_3, the Archive job checks the target where a loaded main program is to be restored and makes sure that the identity of the archive job has enough access to the enclosing world and the existing object to create a new version of the object, *before* it demotes the existing object to archived state. If the job does not have appropriate access then the Copy command fails with a detailed error message explaining the problem. PRS number 9123690-0158-2 and CSR number 5079.

7.3.3. Archiving Searchlists

In previous releases, the Archive.Copy command failed if you attempted to copy a searchlist to another machine on the network and the entries in the searchlist could not be resolved on the target machine. The Copy command failed with a misleading error message saying that the subclass of the restored object could not be set to searchlist.

In D_12_7_3, the Archive.Copy command writes a message saying that the searchlist could not be properly restored on the target machine because it contains entries that are not resolvable on the target machine.

The searchlist is copied as a text file onto the target machine. You can then log into the target machine and edit the text file so that all the entries are resolvable and then execute the command Search_List.Revert to make the text file into a valid search list. PRS number 9123690-0183-5 and CSR number 6426.

7.3.4. Archiving Ventures

When used on ventures that contain string fields, the Archive.Copy command no longer inserts extra quotation marks into those strings. PRS number 1005010-Etoi-Ksch.

7.3.5. Archiving to Tape Across a Network

In a recent release of the Environment the capability to do an Archive.Save to a remote machine's tape drive across the network was added. However, there was a bug in this capability such that the tape would not be unloaded from the remote tape drive when the Archive.Save was finished. This has been fixed and the tape is unloaded unless otherwise specified. PRS number 747500-Gato-Rjg.

7.3.6. Archive Server and Session Tokens

The Archive Server is a job that is started each time the system boots. This job exists to process requests from other R1000's that are attempting to Archive.Copy or Archive.Restore objects onto the current R1000 across the network. The Archive Server also processes Remote.Run jobs that are sent from other R1000's on the network. The Archive Server works in the following way:

1. It receives an incoming request to process.
2. It spawns another Archive.Server job, which waits for another request from a remote R1000.
3. It (the original Archive Server job) takes the initial remote request and processes it.



In previous releases, the Archive Server jobs all ran under the same session (!Machine.Network_Public_Session). This caused a problem because some remote requests require that the Archive Server consume a layered-product token to accomplish the request. Tokens are session-based and are not released until the session is logged out or terminated. Unfortunately, since there is always a Archive Server job running, the session never terminates, so the token was never released.

In this D_12_7_3 release, the Archive Server jobs cycle through seven new sessions. When an Archive Server job finishes, the session terminates and the token is released to be scavenged when it is needed.

The problem described above was only a problem on token-based R1000's but the changes will appear on all R1000s so you should not be surprised when you see this different behavior. The new behavior you will notice is as follows:

- The output from the What.Users command contains two or more entries for the identity Network_Public:
 - One entry contains all of the jobs that were traditionally run under the session !Machine.Network_Public_Session, such as the Print Queue Server and the Mail Distribute Server.
 - The additional entries for the identity Network_Public each contain only one job, an Archive Server job.

To determine the session under which a particular job (such as an Archive Server job) is running, see the new Show_Session_Of_Job command (section 6.4).

- The new world !Machine.Network_Public_Archive_Server_Sessions contains objects for seven new sessions to be used by the Archive Server. These sessions are called Network_Public_Session_1 through Network_Public_Session_7.
- The first time the system is booted after D_12_7_3 has been installed, you will see messages that indicate that the new sessions have been created. It will look something like the following:

```
19:01:29 !!! Predefined_Users User_create_failed : Public: User
already exists
19:01:30 !!! Predefined_Users Prohibit_login_failed Delete status
for Public S_1 : NAME_ERROR
19:01:30 !!! Predefined_Users User_create_failed : Network_Public:
User already exists
19:01:30 !!! Predefined_Users Prohibit_login_failed Delete status
for Network_Public.S_1 : NAME_ERROR
19:01:30 !!! Predefined_Users User_create_failed : Operator: User
already exists
19:01:30 --- Predefined_Users Created_group : Privileged
19:01:30 --- Predefined_Users Created_group : Mailer
19:01:31 --- Predefined_Users Created_group : Spooler
19:01:31 !!! Predefined_Users User_create_failed : Rational: User
already exists
19:01:31 --- Boot Running "!Machine.Initialization".Start
19:01:31 --- Predefined_Users Created_group : System
19:01:31 --- Boot Running Destroy deleted Ada/Link objects
19:01:33 +++ Predefined_Users Created_Session
!Machine.Network_Public_Session
```



```

19:01:33 +++ Predefined_Users Created_Session
!Machine.Public_Session
19:02:55 +++ Predefined_Users Created_World !Machine.
Network_Public_Archive_Server_Sessions
19:02:55 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_1
19:02:56 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_1
19:02:56 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_2
19:02:58 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_2
19:02:58 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_3
19:02:59 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_3
19:02:59 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_4
19:03:00 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_4
19:03:00 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_5
19:03:01 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_5
19:03:01 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_6
19:03:02 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_6
19:03:03 +++ Predefined_Users Attempting_to_create_session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_7
19:03:03 +++ Predefined_Users Created_Session !Machine.
Network_Public_Archive_Server_Sessions.Network_Public_Session_7

```

7.4. Compilation Changes

In previous releases, status messages produced during the compilation of objects contained the version of the object being compiled (e.g., !Users.Geb.Bug_Regressions.Pass_Deferred_Private'Body'V(1).). The 'V(n)' part was determined to be extraneous and sometimes confusing. In D_12_7_3, compilation messages contain only the full path name of the object (e.g., !Users.Geb.Bub_Regressions.Pass_Deferred_Private'Body.). PRS number 2911238-Jude-Geb.

7.4.1. Installing Units

In previous releases, if a package contained both a *with* clause referencing package System and a variable named System, then the variable declaration would hide the *with* to package System and constructs such as representation specifications would not code, saying that there was no visibility to package System. In D_12_7_3, the R1000 compilation system locates the "with System" clause directly, without relying on visibility. PRS number 3078226-Gato-Mboy.

Renaming a field in a constant no longer fails with an error indicating that something was not fully constrained. In D_12_7_3, constants are fully constrained and fully renamable. PRS



number 3108823-Sier-Geb.

The semantic checker now flags as ambiguous a series of derived types and procedures that use those types as parameters. PRS number 4711901-Rati-Pbk.

In previous releases, if the completion type of a forward declaration was in a package body and that type was a discriminant record type, the semantic checker did not always find semantic errors in the completion record type. In D_12_7_3, two-part records are semantically checked in the same manner regardless of whether they exist in separate declaration contexts or the same declaration context. PRS number 5844095-Aria-Sbur.

In accordance with LRM 7.4(4), the semantic checker no longer allows the declaration of a deferred constant for a private type that has been declared in another unit. (LRM 7.4(4) states that "a deferred constant declaration and the declaration of the corresponding private type must both be declarative items of the visible part of the same package.") PRS number 6370093-Etoi-Phl.

In D_12_7_3, the semantic checker now flags as ambiguous a function call that returns an array type when:

- The function has a single integer parameter, and
- The declaration of the integer parameter includes a default value (such as **function x (Int : Integer => 6) return array**), and
- That integer parameter is specified without an explicit assignment. For example, **x(2)** is ambiguous, but **x(Int => 2)** is not.

x(2) should be flagged as ambiguous because it does not specify whether the argument is to be the parameter passed into the function or an index into the returned array type. In previous releases, the semantic checker assumed that it was an index into the return array and the value for the parameter was given its default value. PRS number 7762494-Rati-Pbk.

The semantic checker now handles representation specifications for Long_Integer types. In previous releases, the semantic checker would sometimes fail with a Numeric_Error (Overflow). PRS number 8391944-Gato-Gbd.

In accordance with Ada Issue AI-00438/09, the semantic checker now allows functions that rename enumeration literals, rather than flagging them as not static. (Ada Issue AI-00438/09 states: "A function call whose function name denotes an enumeration literal, a predefined operator, or a function that is a language-defined attribute of a static subtype is allowed as a primary in a static expression if the value of each parameter is given by a static expression.") PRS number 8480408-Gato-Gbd.

The semantic checker no longer flags as erroneous the following situation:

- A generic package contains a type declaration and a subtype declaration of that type, and
- There is an instantiation of the generic, and
- There is a procedure that *withs* that instantiation and contains declared variables of both the type and the subtype, and
- There is an attempt to assign those variables to one another.

PRS number 9123690-0184-6 and CSR number 6625.



Improved Messages:

In D_12_7_3, the semanticist no longer generates a garbled message with the real error explanation when a semantic error occurs due to an invalid representation specification. Only the real error message is shown. PRS number 12862-Star.

In previous releases, if a record type contained a field that was not semantically correct or not defined and that type had a representation specification, then the rep spec would be flagged with a non-meaningful error message. In version, this has been fixed and the actual problem is flagged and described correctly. PRS number 3496838-Gato-Rjg.

In D_12_5_0 and later releases, the error message was formatted badly when a semantic error was flagged indicating that a call was made to a nonexistent procedure or function. The format has now been fixed. PRS number 9072739-Shei-Swb.

7.4.2. Coding Units

The R1000 code generation now generates correct code when a type completion of a forward declaration is a derived type. PRS number 3032909-Cook-Swb.

In previous releases, the R1000 code generator would fail to code a subprogram containing an enumerated type of characters that had a representation specification when a string literal of a string type was declared with this enumeration type of character. Instead, the code generator raised the internal error **Unimplemented: Element type of string literal has non-static size**. This problem has been fixed and the correct code is generated. PRS number 6969167-Gato-Gbd.

7.4.3. Loading and Executing Units

In previous releases, a program would raise a `Type_Error` when executing a `Return` statement if the function executing the `Return` contained a variable that was code-optimized incorrectly. The locally declared variable that was being returned would get popped from the control stack when the `Return` was executed, and its type was lost. This has been fixed and the type of the variable being returned is preserved until the return is complete. PRS number 4290910-Sier-Geb.

Unhandled exceptions from the Environment that result from errors in user programs now contain the string name of the exception. In the D_12_5_0 release, there were some cases in which the Environment displayed the internal numeric representation for the exception rather than the string name. PRS number 9031863-Gato-Gbd.

7.4.4. Incremental Compilation

Incrementally adding a type completion now produces the same DIANA as recompiling the object in batch mode. In previous releases, if a type completion of a forward declaration was incrementally inserted into a package, the DIANA was left in an inconsistent state and did not indicate that the forward declaration had been completed. PRS number 426364-Shei-Jst.



7.5. Miscellaneous Library Management Changes

The Library.Space command no longer fails with a **Numeric_Error (Overflow)** message. PRS number 12612-Star.

The logs generated by the Access_List.Set and Access_List.Set_Default commands now echo the value of the Response parameter. PRS number 523289-Sier-Geb.

Delete access is no longer required to the destination world to copy or move an Ada package specification and body in a single operation. This change fixes a problem introduced in D_12_5_0 in which attempts to copy or move Ada unit bodies and specifications to a world for which the user did not have Delete access copied (or moved) the body but failed to copy (or move) the specification, saying that Write access was required. PRS number 9123690-0194-1.

7.6. Changes to Crash Analysis

The D_12_7_3 release of the Environment includes several major changes to crash analysis. 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 Rational 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.

For information about how to respond to the crash of a D_12_7_3 system, see Appendix B, "Diagnostic Crash Procedures for Release D_12_7_3" in the *System Manager's Guide* delivered with this release. The following sections describe the Failure Reboot and Automatic Notification features. Descriptions of these features are also found in the new *System Manager's Guide*.

7.6.1. Failure Reboot

In previous releases, the machine, when properly configured, would automatically reboot following normal shutdowns. If the machine crashed due to a hardware or software error, it would require the operator to respond to prompts for the machine to boot.

In D_12_7_3, the system 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.



Four new procedures in package `!Machine.Dfs'Spec_View.Units.Dfs` control the Failure Reboot feature (see section 6.3 for descriptions):

- `Reboot_On_Failure`
- `Reboot_On_Failure_Interval`
- `Quiesce_Reboot_On_Failure`
- `Reboot_On_Failure_Settings`

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 Rational 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 subsection). 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 Rational.

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.

7.6.2. Automatic Notification

After a system failure occurs, the system produces an analysis of the failure and writes it to the system error log. In addition, system managers can specify additional distribution of the analysis output in a new control file. 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 description 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 Rational. By default, this field is blank.

7.7. Miscellaneous Changes to System Management

The Message.Send command no longer sends unnecessary error messages to the sender when wildcard characters are used in the Who parameter. In previous releases, if the Who parameter contained wildcards, an error message would be generated to the sender, but the message would be sent properly. PRS number 9123690-0180-9 and CSR number 6106.

The Abbreviations.Print command now deletes the temporary file it creates when it is used to print a mail message or other image that does not correspond to an Environment object. These temporary files are located in !Machine.Temporary, and in previous releases, were not deleted automatically. PRS number 1033878-Gilg-Cmd.

7.8. Changes to CMVC

7.8.1. Make_Controlled and the State Directory

The Cmvc.Make_Controlled command now works properly if the State directory of a view and/or the State directory's contents are specified. In particular:

- Attempts to control the State directory now fail with the message **The State directory cannot be controlled.**
- Attempts to control objects in the State directory succeed without generating error messages.

PRS number 6289037-Gilg-Cmd.

7.8.2. Destroy_View and Frozen Objects

The Cmvc.Destroy_View now unfreezes access-control files in the Configurations.Configuration_Name_State directory of the view, if necessary. PRS number 6626937-Flor-Jst.



7.8.3. Naming Conflicts when Copying or Building Views

In D_12_7_3, if you attempt to use `Cmvc.Copy` (or a similar CMVC command) to copy a view in which there is a naming conflict between a unit in the view and a unit in the view's imports, the Copy command maintains the internal link to the local unit and generates a warning message describing the naming conflict. In previous releases, if the view to be copied contained such a naming conflict, the Copy command failed, saying that the imported link conflicted with the existing internal link. All copying to that point was be lost. PRS number 2778001-Sier-Geb.

Similarly, in D_12_7_3, if you attempt to build a view from a configuration in which there is a naming conflict between a unit in the configuration and a unit in the new view's imports, the command maintains the internal link to the local unit and generates a warning message describing the naming conflict. The build completes successfully. PRS number 9496712-Wood-Ken.

7.8.4. Default Parameter Values for `Make_Spec_View`

The default value of the `Level` parameter of the `Cmvc.Make_Spec_View` procedure has been changed from zero (0) to one (1). The new specification is:

```
procedure Make_Spec_View
(From_Path      : String           := "<CURSOR>";
 Spec_View_Prefix : String           := ">>PREFIX<<";
 Level          : Natural           := 1;
 View_To_Modify  : String           := "";
 View_To_Import : String           := "<INHERIT_IMPORTS>";
 Only_Change_Imports : Boolean      := True;
 Remake_Demoted_Units : Boolean     := True;
 Goal           : Compilation.Unit_State := Compilation.Coded;
 Comments       : String           := "";
 Work_Order     : String           := "<DEFAULT>";
 Volume        : Natural           := 0;
 Response      : String           := "<PROFILE>");
```

7.8.5. Consecutive Calls to Release

In previous releases, consecutive calls to the `Cmvc.Release` command produced non-sequential release numbering. This has been fixed and release numbers are generated sequentially. CSR number 7334.

7.8.6. `Cmvc_Access_Control`

The `Cmvc_Access_Control.Check` command no longer fails if it is used on and attempts to update an entry in a full access list (one that contains seven entries). In previous releases, the Check command failed because it tried to add the correct entries before removing the incorrect entries; thus, it would report that there were too many entries and would fail. The Check command now removes the incorrect entries before added the correct entries, and, thus, works successfully. PRS number 3461738-Gilg-Cmd.

When the name of a subsystem is specified to the `View_Or_Subsystem` parameter of the `Cmvc_Access_Control.Check` command, the command now checks the subsystem itself and all



the views in that subsystem. In previous releases, the command would only check the subsystem. PRS number 7842145-Gilg-Geb.

Users that belong to a group with Developer access to a subsystem can now create new objects within that subsystem. In previous releases, only members of groups with Owner access were permitted to create objects within a subsystem. PRS number 8819874-Capi-Rati.

7.8.7. Log Messages

The `Cmvc.Replace_Model` command has been made more robust and error messages have been improved. In previous releases, the `Replace_Model` command would sometimes fail if the `View.State.Last_Release_Name` file was not formatted correctly and could not be parsed. In such cases, the error messages generated were not helpful. The error messages have been rewritten and the command now handles many minor errors that would have caused it to fail in the past. PRS number 9123690-0153-5 and CSR number 4843.

The messages generated by the `Cmvc_Maintenance.Check_Consistency` command now include the actual CMVC database object, rather than just the subsystem, when the command fails because the database for a subsystem cannot be opened. PRS number 5641327-Sier-Geb.

7.8.8. Work Orders

In previous releases, executing `Common.Definition` on a very large work order failed with a `Storage_Error` from the work order editor. In `D_12_7_3`, the work order editor has been improved to handle larger work orders. PRS number 9123690-0199-1 and CSR number 7751.

7.9. Changes to Activity_Implementation

The `!Implementation.Activity_Implementation` package and its contents are not intended for use by customer-defined programs; they are for use by Environment-defined subprograms and packages. Information about changes to `Activity_Implementation`, however, are provided on the chance that some sites may have found it necessary to build tools against it.

The `Default_Handle` function has been removed from the specification of package `Activity_Implementation`. This function was never meant to be visible and generated unhandled exceptions. PRS number 6527216-Gato-Bas.

In the `D_12_6_5` release, the specification of package `Activity_Implementation` was out of sync with the implementation, resulting in `!Lrm.System.Operand_Class_Error` messages being generated by some procedures. This has been fixed, and the procedures in package `Activity_Implementation` should work as advertised. PRS number 9123690-0174-1 and CSR number 6022.

7.10. Networking Changes

In previous releases, some Telnet operations were extremely slow on the Series 400 processor. The slow performance was caused by the CMC Ethernet Controller (new on the 400's) interpreting the bit pattern `16#FF#` as a packet terminator and sending one packet for each such bit pattern. This behavior lead to much unneeded packet traffic and even timeouts on the



network. This problem has been fixed; Telnet performance is back to normal. PRS number 3443626-Suno-Ken.

Connections from an R1000 to a UNIX workstation via Ftp.Connect or Telnet.Connect have been made more reliable. In previous releases, the R1000 sometimes echoed the login request to the UNIX workstation when it should not have. The UNIX workstation could not handle this echo, so the connections used to fail. PRS number 9123690-0187-0.

In previous releases, Series 200 or 300 systems would sometimes crash with a !Lrm.System-Assertion_Error from the Tcp_Ip_Driver software. In particular, the system would crash if:

- An application called Transport.Connect to make a passive connection on the network, and
- This connection timed out before completion, and
- The application immediately tried the passive connection again.

This problem was determined to be a timing problem between the R1000 Kernel software and the Network board in the R1000. The Kernel software has been updated to handle this short timing discrepancy and the crash no longer occurs. This was never a problem with the Series 400 because it has a different Network board. PRS number 9057421-Back-Rfg.

7.11. LRM Interface Changes

The D_12_7_3 release of the Environment includes a new release of the LRM Interface. If you are already running Rational Design Facility release 7_2_1, this change will not affect you; the new LRM Interface was delivered with that release.

If you are upgrading to the LRM release included in D_12_7_3, note that changes made to the Declaration_Kinds, Attribute_Designator_Kinds, and Pragma_Kinds enumerations may result in compilation or runtime errors (see subsections 7.11.20 and 7.11.24 for details).

The procedure Find_Lrm_Change_Candidates, located in the Pdl_Commands subsystem, can be used to assist in identifying potential compatibility problems that are not caught by the compiler.

7.11.1. Discriminated Type Function Changes

Type_Information.Is_Discriminated and Type_Information.Discriminants have been changed to work on generic formal types, private types, limited private types and incomplete type. To handle incomplete types, these functions now accept type declarations as well as type definitions.

```
function Is_Discriminated
  (A_Type : Type_Definition_Or_Declaration) return Boolean;
-- This function applies to private, limited private, incomplete or
-- record types. It returns True if this type has discriminants.
-- It may be applied to type declaration to handle the case of
-- incomplete types.
```

```
function Discriminants (A_Type : Type_Definition_Or_Declaration)
  return Discriminant_Iterator;
-- Returns a list of discriminants of the type. These elements may
```



```
-- then be manipulated with the functions provided in package
-- Declarations.
```

Note that the name of the formal parameter has been changed from Record_Type to A_Type. This is because the former was a misnomer. Record types are not the only types with discriminants.

7.11.2. New Aggregate Range Function

The following function has been added to package Names_And_Expressions.

```
function Aggregate_Range (An_Aggregate : Expression)
  return Discrete_Range
-- For an array aggregate, returns a range specifying the bounds of
-- the aggregate.Associations
```

7.11.3. Declarations.Kind Change

Declarations.Kind has been changed to be consistent with the kind operations of other packages. That is, it now checks the element's major kind and returns Not_A_Declaration if the major kind is not a declaration.

7.11.4. New Associations Package

A new package called Associations is now available. This package may be used to analyze procedure, entry and function calls, generic instantiations and pragma arguments. The Associations package exports the following services:

```
type Association_Kinds is (Named_Association,
                          Positional_Association,
                          Defaulted,
                          Not_An_Association);

function Association_Kind
  (An_Association : Association) return Association_Kinds;
-- Returns the kind of an association.

function Formal_Parameter (An_Association : Association)
  return Identifier_Definition;
-- Returns the identifier of the formal name for the given
-- association. This function tries hard to return an identifier
-- definition. However, in the case of a pragma argument, only a
-- reference can be returned.

function Actual_Parameter
  (An_Association : Association) return Name_Expression;
-- Returns the actual name or expression for the given association.
```



7.11.5. Lexical Typing of Iterators

Lexical typing of iterators has been added. Lexical typing is the practice of using subtype name to reflect the kind of major elements that they can contain. The following declarations have been added to Ada_Program:

```

subtype Association_Iterator is Element_Iterator;
subtype Choice_Iterator is Element_Iterator;
subtype Compilation_Unit_Iterator is Element_Iterator;
subtype Context-Clause-Or-Pragma_Iterator is Element_Iterator;
subtype Declaration-Or-Context-Clause-Or-
  Representation-Clause-Or-Pragma_Iterator is Element_Iterator;
subtype Expression_Iterator is Element_Iterator;
subtype Name_Iterator is Element_Iterator;
subtype Pragma_Iterator is Element_Iterator;
subtype Representation-Clause_Iterator is Element_Iterator;
subtype Statement-Or-Pragma_Iterator is Element_Iterator;
subtype Type_Definition_Iterator is Element_Iterator;
-- Note that some of the iterators can mix items of different major
-- kinds. Their name attempts to convey this information. For
-- instance a declarative part can contain, besides declarations,
-- context clauses (viz. use clauses), representation clauses or
-- pragmas.

```

Many subprogram specifications have been changed, in all packages, to take advantage of the new names. A number of local iterator names have also been introduced to shorten the identifiers or to denote specific constructs.

In Declarations:

```

subtype Declarative_Part_Iterator is
  Ada_Program.Declaration-Or-Context-Clause-
    Or-Representation-Clause-Or-Pragma_Iterator;

subtype Subprogram_Formal_Parameter_Iterator is
  Ada_Program.Element_Iterator;

subtype Generic_Formal_Parameter-Or-Pragma_Iterator is
  Ada_Program.Element_Iterator;

```

In Names_And_Expressions:

```

subtype Aggregate_Component_Iterator is Ada_Program.Element_Iterator;

```

In Representation_Clauses:

```

subtype Record_Component-Clause-Or-Pragma_Iterator is
  Ada_Program.Element_Iterator;

```

In Statements:

```

subtype Declarative_Part_Iterator is
  Ada_Program.Declaration-Or-Context-Clause-
    Or-Representation-Clause-Or-Pragma_Iterator;

```



```
subtype Statement_Part_Iterator is
  Ada_Program.Statement_Or_Pragma_Iterator;

subtype If_Statement_Arm_Iterator is Ada_Program.Element_Iterator;

subtype Case_Statement_Alternative_Iterator is
  Ada_Program.Element_Iterator;

subtype Exception_Handler_Arm_Iterator is Ada_Program.Element_Iterator;

subtype Select_Alternative_Iterator is Ada_Program.Element_Iterator;
```

In Type_Information:

```
subtype Declarative_Part_Iterator is
  Ada_Program.Declaration_Or_Context_Clause_Or_
  Representation_Clause_Or_Pragma_Iterator;

subtype Discrete_Range_Iterator is Ada_Program.Element_Iterator;

subtype Discriminant_Association_Iterator is
  Ada_Program.Element_Iterator;

subtype Discriminant_Iterator is Ada_Program.Element_Iterator;

subtype Record_Component_Or_Pragma_Iterator is
  Ada_Program.Element_Iterator;

subtype Variant_Or_Pragma_Iterator is Ada_Program.Element_Iterator;
```

7.11.6. Compilation Unit Pragmas

The following function was added to package `Compilation_Units`:

```
function Attached_Pragmas
  (To_Compilation_Unit : Compilation_Unit) return Pragma_Iterator;
-- Returns the list of pragmas attached to a compilation unit. Only
-- those pragmas that follow the compilation unit are returned here.
-- Pragmas that precede the compilation unit are part of the context
-- clause.
```

7.11.7. Operations on Generic Instantiations

In Declarations, the following operations have been modified so that, when applied to a generic instantiation, they analyze the un-rooted tree to report information of the instance:

```
function Visible_Part_Declarations
  (Package_Specification : Package_Declaration)
  return Declarative_Part_Iterator;
-- Returns a list of all declarations, representation
-- specifications, and pragmas in the visible part of a package in
-- the order of appearance. When applied to a package
-- instantiation, this operation yields the instance's visible
```



```
-- declarations.
```

```
function Private_Part_Declarations
  (Package_Specification : Package_Declaration)
  return Declarative_Part_Iterator;
-- Returns a list of all declarations, representation
-- specifications, and pragmas in the private part of the package in
-- order of appearance. When applied to a package instantiation,
-- this operation yields the instance's private declarations.
```

```
function Subprogram_Parameters (Subprogram_Or_Entry : Declaration)
  return Subprogram_Formals_Iterator;
-- Returns an ordered list of formal parameter declarations.
-- Use IS_INITIALIZED and INITIAL_VALUE to query
-- the information related to the presence of the default parameter
-- initialization, and use TYPE_MARK to obtain the parameter type mark.
-- When applied to a subprogram instantiation, this operation yields
-- the instance's parameters.
```

```
function Return_Type (Of_Function : Function_Declaration)
  return Name_Expression;
-- Returns the name expression of the return type, selectors in
-- NAMES_AND_EXPRESSIONS can then be used to extract more information.
-- When applied to a function instantiation, this operation yields
-- the instance's return type.
```

In Declarations, the following subprograms have been added or modified to improve support of generic instantiations:

```
function Generic_Unit_Declaration
  (Generic_Instantiation_Or_Unit_Declaration : Declaration)
  return Declaration;
-- Returns the declaration of the generic unit being instantiated.
```

```
function Generic_Instantiation_Parameters
  (Generic_Instantiation : Declaration) return Association_Iterator;
-- Returns an ordered list of parameter associations of a generic
-- instantiation. The operations defined in package ASSOCIATIONS
-- can be used to decompose them.
```

```
function Generic_Actual_Parameters (Generic_Instantiation : Declaration)
  return Association_Iterator
  renames Generic_Instantiation_Parameters;
-- Use of this form is discouraged.
```

7.11.8. Functions and Enumeration Literal Renames

In Names_And_Expressions, the following comments have been added to document the fact that a function call may be a renaming of an enumeration literal:



```
-- FUNCTION CALLS
-- Note that references to enumeration literals renamed as functions
-- are treated as genuine function calls
```

7.11.9. Constants versus Variables

In order to be able to distinguish between constant and variable objects, the following function was added to Names_And_Expressions:

```
function Is_Constant (A_Name : Name) return Boolean;
-- Returns True if the given name is constant. The name must be
-- of a syntactic form suitable for the left hand side of an
-- assignment (ie. not an attribute, a character, etc.).
```

7.11.10. Task Entries versus Subprograms

In consistency with the LRM 6(2), Declarations.Is_Subprogram has been changed to return False on an entry.

7.11.11. Comments in Declaration Names

Declarations.Name no longer return same line comments. It is the equivalent of applying Ada_Program.String_Name to the result of the Declarations.Identifiers.

7.11.12. New Labels Function

The following function has been added to package Statements:

```
function Labels (A_Statement : Statement) return Name_Iterator;
-- Returns an iterator on the names of the labels of a statement. A
-- statement can have several labels.Declarations
```

7.11.13. Limited Private Generic Formal Parameters

The enumeration type in package Declarations that defines generic parameter kinds was changed to handle limited private formal types. The behavior of function Generic_Parameter_Kind has been changed accordingly.

```
type Generic_Parameter_Kinds is (Subprogram,
                                Object,
                                Private_Type,
                                Limited_Private_Type,
                                Discrete_Type,
                                Integer_Type,
                                Floating_Point_Type,
                                Fixed_Point_Type,
                                Array_Type,
                                Access_Type,
                                Not_A_Generic_Parameter);
```



7.11.14. Change to Renaming Declarations Function

Declarations.Is_Renaming_Declaration has been changed so that, if given an identifier definition, it goes to the corresponding declaration. This makes it consistent with all of the Is_@ functions in this package.

7.11.15. Renamed Declarations

The first function below has been added to Declarations. The second has been documented and restricted to avoid returning meaningless information:

```
function Renamed_Name
  (A_Declaration : Declaration) return Name_Expression;
-- Returns the name of the entity being renamed. It can be a simple
-- name, an operator symbol, an indexed component, a slice, a
-- selected component or an attribute.

function Renamed_Declaration
  (A_Declaration : Declaration) return Declaration;
-- If applied to the renaming of a simple name, an operator symbol
-- or an expanded name, returns the name's declaration. Returns nil
-- element otherwise. Use of this function is discouraged.
```

7.11.16. Subunits Without Bodies

Declarations.Unit_Body now returns Ada_Program.Nil_Element when passed the specification of a subunit for which there is no body object in the library rather than raising the exception Inappropriate_Program_Element.

7.11.17. Subunits Without Specs

When Declarations.Specifications is passed a subunit stub or body identifier, it will now return the stub declaration, regardless of whether or not the subunit has a specification.

Previously, this function returned the specification, if one existed for the subunit. In cases where a the subunit had no spec, the stub was returned when passed the identifier from the Is_Separate clause but a Nil_Element was returned when passed the identifier of the subunit body.

This behavior is consistent with LRM 6.3(3). This is also the change with minimal impact, since few programs should depend on the Nil_Element. Note however that this behavior is somewhat inconsistent with Is_Spec, which returns False on a stub.Declarations.Specifications

7.11.18. Declarations.Is_Initialized Change

Declarations.Is_Initialized has been changed to test the nature of its third child, returning True if it is a true initial value, and raising Inappropriate_Program_Element if it is a renaming.



7.11.19. New Function for Incomplete Types

The following function has been added to the Declarations package.

```
function Full_Type_Declaration
  (Type_Declaration_Or_Id : Declaration) return Type_Declaration;
-- Given the declaration of an incomplete type, returns the
-- corresponding full type declaration. A nil element is returned
-- if the full type declaration is not yet compiled. NOTE: this is
-- the identity function if given a non-incomplete type declaration.
```

7.11.20. Implementation-Dependent Attributes and Pragmas

Attribute handling in Names_And_Expressions has been changed to better handle implementation specific attributes. Attribute handling now looks like:

```
type Attribute_Designator_Kinds is (
  Address_Attribute,
  Aft_Attribute,
  Base_Attribute,
  Callable_Attribute,
  Constrained_Attribute,
  Count_Attribute,
  Delta_Attribute,
  Digits_Attribute,
  Emax_Attribute,
  Epsilon_Attribute,
  First_Attribute,
  First_Bit_Attribute,
  Fore_Attribute,
  Image_Attribute,
  Large_Attribute,
  Last_Attribute,
  Last_Bit_Attribute,
  Length_Attribute,
  Machine_Emax_Attribute,
  Machine_Emin_Attribute,
  Machine_Mantissa_Attribute,
  Machine_Overflows_Attribute,
  Machine_Radix_Attribute,
  Machine_Rounds_Attribute,
  Mantissa_Attribute,
  Pos_Attribute,
  Position_Attribute,
  Pred_Attribute,
  Range_Attribute,
  Safe_Emax_Attribute,
  Safe_Large_Attribute,
  Safe_Small_Attribute,
  Size_Attribute,
  Small_Attribute,
  Storage_Size_Attribute,
  Succ_Attribute,
  Terminated_Attribute,
```



```

Val_Attribute,
Value_Attribute,
Width_Attribute,
Not_A_Predefined_Attribute);

```

```

function Attribute_Designator_Kind
  (Attribute : Name) return Attribute_Designator_Kinds;
-- Returns the kind of an attribute. If the attribute is
-- implementation-specific, Not_A_Predefined_Attribute is returned.

```

```

function Attribute_Designator_Name (Attribute : Name) return String;
-- This is the preferred way to analyze an implementation-specific
-- attribute. It returns an uppercase string for the attribute
-- simple name.

```

```

function Attribute_Designator_Name (Attribute : Name) return Name;
-- The Simple_Name returned here is only intended for use by
-- ADA_PROGRAM.STRING_NAME. Use of this function is discouraged.

```

In addition, package Pragma has been changed to better handle implementation specific pragmas. It now looks like:

```

function Is_Predefined (A_Pragma : Pragma_Usage) return Boolean;

```

```

type Pragma_Kinds is (Controlled,
  Elaborate,
  Inline,
  Interface,
  List,
  Memory_Size,
  Optimize,
  Pack,
  Page,
  Priority,
  Shared,
  Storage_Unit,
  Suppress,
  System_Name,
  Not_A_Predefined_Pragma);

```

```

Unknown : constant Pragma_Kinds := Not_A_Predefined_Pragma;

```

```

function Kind (A_Pragma : Pragma_Usage) return Pragma_Kinds;
-- Returns the kind of a pragma. Returns Not_A_Predefined_Pragma on
-- implementation-specific pragmas.

```

```

function Name (A_Pragma : Pragma_Usage) return String;
-- Returns the uppercase simple name of any pragma. This is the way

```



```
-- to analyze implementation-specific pragmas.
```

```
function Arguments (A_Pragma : Pragma_Usage)
  return Association_Iterator;
-- Returns a list of the arguments to a pragma. Operations from
-- package ASSOCIATIONS can be used to decompose them.
```

Note that R1000-specific pragmas have been removed from the Pragma_Kinds. The names of these and other implementation specific pragmas can be used to analyze them.

7.11.21. Return Type of Function Instantiations

Declaration.Return_Type has been changed to yield the instance return type when applied to a function instantiation. Note that this return type will be a subtype renaming the true actual type.

7.11.22. Block and Loop Names

The following functions have been added to package Statements in order to query for loop and block names.

```
function Is_Named_Statement (A_Statement : Statement) return Boolean;
-- Returns true if applied to a loop or block that has a name.
```

```
function Statement_Name (A_Statement : Statement) return Name;
-- Returns the name of a block or loop. Returns Nil_Element if not
-- a block or loop, or if no name is present.
```

7.11.23. Enumeration Literal Queries

In Names_And_Expressions, Is_Literal, Position_Number and Representation_Value have been changed to work on an enumeration literal specifications. Previously, they worked only on a reference to an enumeration literal.

7.11.24. Record Discriminant and Component Declarations

The following literals have been added to type Declaration_Kinds in order to differentiate record components from variables.

```
A_Discriminant,
A_Record_Component,
```

Ada_Program.Kind and Declarations.Kind have been modified to return these new values when appropriate. Declarations of these kinds can be analyzed by Is_Initialized, Initial_Value, and Object_Type.



8. Documentation

The D_12_7_3 release of the Rational Environment is accompanied by new and updated printed and online documentation.

8.1. Printed Documentation

This D_12_7_3 release is accompanied by two printed manuals:

- An up-to-date *System Manager's Guide* for the Rational R1000 Software Development System, Series 200, 300, and 400. This document has been tested for accuracy with the D_12_6_5 release, reorganized, and presented in a new, easier-to-read format with red tabs marking emergency procedures. The *System Manager's Guide* is product number 4000-00142.
- The *Rational Access User's Guide*, for use with the Rational Access user interface. This document is product number 4000-00722.

You should receive one copy of each manual for every system under an active support contract. If you have not received your new documentation or if you would like to order additional documentation, please contact your Rational representative.

8.1.1. Correction to *Quick Reference for Parameter-Value Conventions*

On the second page of Appendix F, "Quick Reference for Parameter-Value Conventions," in the *System Manager's Guide*, the table describing special names for default objects describes the special name "<ACTIVITY>" as resolving to the "default activity for the current session." This is incorrect. "<ACTIVITY>" resolves to the current activity, which is the activity named in the job response profile for the current job. Unless you have explicitly set the activity for the job, the activity is the same as the default activity for the current session.

This correction also applies to the versions of the "Quick Reference for Parameter-Value Conventions" in the Session and Job Management, Library Management, and System Management Utilities books of the *Rational Environment Reference Manual*.

8.2. Online Documentation

This D_12_7_3 release of the Environment includes new or updated online documentation for:

- Rational's online help system. To display this information, press [Help on Help] or execute **What.Does ("Help_On_Help")**.
- The packages described in the Debugging (DEB) book of the *Rational Environment Reference Manual*:
 - Package Debug
 - Package Debug_Maintenance
 - Package Debug_Tools
- The packages described in the Project Management (PM) book:
 - Package Activity



Rational Environment Release Information

- Package Check
- Package Cmvc
- Package Cmvc_Access_Control
- Package Cmvc_Hierarchy
- Package Cmvc_Maintenance
- Package Work_Order

An effort has been made to bring the online documentation for the Debugging and Project Management packages up-to-date with the D_12_7_3 release of the Environment. The new online help for these packages also contains parameter-level information, examples, and cross references previously not available.

This D_12_7_3 release also includes online PostScript versions of the keyboard overlays for the Rational X Interface. These overlays are located in !Machine.Editor_Data.Keyboard_Overlays. To print an overlay from the Environment, specify the print option Original_Raw.

9. Training

Rational is currently in the process of updating the standard Environment training courses to reflect this D_12_7_3 release of the Environment and to incorporate the paradigm changes introduced by the Rational Access user interface. Rational will publicize the availability of these courses when they are complete.



Appendix A

Problem Reports Closed in D_12_7_3

This appendix lists the software problem reports and customer-service requests closed by the D_12_7_3 release.

Table A-1 lists the problem reports fixed by D_12_7_3. It includes the Problem Reporting System (PRS) number of the problem, the Customer Service Request (CSR) number (if applicable), a brief description of the problem, and the section of this release information in which the problem is discussed in more detail (when applicable).

Table A-1 Software Problems Fixed by D_12_7_3

PRS Number	PRS Summary	Section
none	CSR7334: Consecutive calls to Cmvc.Release product non-sequential numbering	7.8.5
10464-Star	What.Locks fails if applied to a deleted unit	7.2
12612-Star	Library.Space gets Numeric_Error	7.5
12728-Star	What.Object does the wrong thing with non-directory objects	7.2
12862-Star	Semantics blows up when rep spec applied to incomplete type	7.4.1
0-0298-0	Archive creates subsystems and worlds within subsystems	7.3.1
167263-Clem-Marl	What.Object on pointy file fails; <IMAGE> is not defined	7.2
426364-Shei-Jst	Incremental operations give different results	7.4.4
523289-Sier-Geb	Access_List.Set@ fails to put Response parameter in log	7.5
747500-Gato-Rjg	Tape Eject behavior inconsistent	7.3.5
1005010-Etoi-Ksch	Archive.[Save,Restore] changes venture fields	7.3.4
1033878-Gilg-Cmd	Abbreviations.Print doesn't delete temporary files	7.7
2115837-Etoi-Phl	Selection sometimes fails in Environment menus	7.1.1
2778001-Sier-Geb	Cmvc.Copy halts when there is a naming conflict	7.8.3
2911238-Jude-Geb	Compilation messages should not have 'V()' in them	7.4
3032909-Cook-Swb	R1000 code bug - type completion for access types	7.4.2
3078226-Gato-Mboy	Overloading of name of package System and variable name causes confusion	7.4.1
3108823-Sier-Geb	Semantics error on renaming a field in a constant	7.4.1
3443626-Suno-Ken	Poor Telnet performance on model 400 processor	7.10
3461738-Gilg-Cmd	Cmvc_Access_Control.Check fails when there are too many extra entries	7.8.6
3496838-Gato-Rjg	Erroneous message in semantically inconsistent code	7.4.1
4290910-Sier-Geb	R1000 code generator generates code that raises Type_Error at run time	7.4.3
4317752-Clem-Marl	What.Object fails when cursor is in selection in text image	7.2



Table A-1 Software Problems Fixed by D_12_7_3 (continued)

PRS Number	PRS Summary	Section
4711901-Rati-Pbk	Ambiguity undetected by semantics	7.4.1
5098655-Mago-Sdj	Problem searching in an image another job has open	7.1.2
5476876-Voya-Phil	What.Locks fails when given a deleted object	7.2
5641327-Sier-Geb	Cmvc_Maintenance.Check_Consistency generates unhelpful message	7.8.7
5844095-Aria-Sbur	Incorrect use of discriminant not detected	7.4.1
6289037-Gilg-Cmd	Cmvc.Make_Controlled generates bogus message when controlling objects in State directory	7.8.1
6370093-Etoi-Phl	R1000 compiler allows deferred constant for private type	7.4.1
6527216-Gato-Bas	Activity_Implementation.Default_Handle blows up in D_12_1_1	7.9
6626937-Flor-Jst	Problem with Cmvc.Destroy_View and frozen access control files	7.8.2
6969167-Gato-Gbd	Unimplemented: String literal has non-static size	7.4.2
7762494-Rati-Pbk	Ambiguous expression not detected by semantic analysis	7.4.1
7842145-Gilg-Geb	Cmvc_Access_Control.Check on a subsystem does not check view	7.8.6
8391944-Gato-Gbd	Can't semanticize record rep clause with long_integer base t	7.4.1
8480408-Gato-Gbd	Renamed enumeration literals can be static	7.4.1
8819874-Capi-Rati	Cmvc_Access_Control: Developer group can't create new objects	7.8.6
8957642-Cook-Swb	Problem with operations on image before editor is done displaying image	7.1.2
8995225-Shei-Jst	Problem with selection in Environment menus	7.1.1
9031863-Gato-Gbd	Exceptions not identified when propagated to Environment	7.4.3
9057241-Back-Rfg	Transport.Connect problem with series 200 and 300	7.10
9072739-Shei-Swb	Semantic error message deterioration in new release	7.4.1
9123690-0153-5	CSR4843: Cmvc.Replace_Model causing error	7.8.7
9123690-0158-2	CSR5079: Archive.Copy corrupts existing Load_Proc on failure	7.3.2
9123690-0174-1	CSR6022: Operand_Class_Error on Activity_Implementation call	7.9
9123690-0180-9	CSR6106: Message.Send problem	7.7
9123690-0183-5	CSR6426: Problem with Archive.Copy and searchlists	7.3.3
9123690-0184-6	CSR6625: Ada compilation discrepancy	7.4.1
9123690-0187-0	Ftp.Connect/Telnet.Connect to UNIX fail due to poor handshaking	7.10
9123690-0194-1	Delete access required for creating Ada units	7.5
9123690-0199-1	CSR7751: Storage error with Definition on work orders	7.8.8



Table A-1 Software Problems Fixed by D_12_7_3 (continued)

PRS Number	PRS Summary	Section
9496712-Wood-Ken	Problem with Cmvc.Build and import conflicts	7.8.3



Table A-2 lists the problem reports that have been investigated and closed because of one of the following reasons:

- The problem could not be reproduced in D_12_7_3.
- The problem has been fixed in a previous Environment release.
- There is not enough information about the problem to proceed.

Table A-2 Not Reproducible in D_12_7_3

PRS Number	PRS Summary
6956-Star	Directory_Uilities.Get_Containing_Subsystem
7689-Star	Archive says "moved" when it means "copied"
8286-Star	Bad messages from Compilation.Demote
8817-Star	Definition on deleted object gave Version_Error
9568-Star	Tape.Write doesn't know when it hits end of tape
9651-Star	Library.Resolve gets unhandled exception
9923-Star	Common.Complete in installed unit
10136-Star	Mail.Send_Message assumes string parameters start at indice = 1
10224-Star	Common.Edit in an activity without a selection fails
10248-Star	Suggest adding more functionality to 'C naming attribute
10391-Star	Tape always unloads even if subsequent read is desired
10491-Star	Directory_Tools.Naming.Ada_Name always returns null string
10723-Star	Should be default sort order switch for window directory
10863-Star	Inconsistency in Directory.Naming
10974-Star	Common.Format fails after expunging mailbox
11048-Star	System_Report.Generate produces meaningless report
11132-Star	Job.Kill crashed machine
11306-Star	Window directory enhancement
12380-Star	Common.Object.Delete of a mailbox failed with Nonexistent_Page_Error
12387-Star	Mail_Internal_Error trying to view unread mail message
12637-Star	Expunge on a non-main mailbox fails
12673-Star	Infinite loop in simplification of 'Width
12722-Star	Syntactic completion not as smart as it should be
12766-Star	Incorrect 'Length
0-0377-0	Strange bug in R1000 compiler
2668045-Wood-Rfg	Implicit exception reported raised at unknown location
2775544-Zebr-Lore	Inappropriate warning regarding limited private types
295703-Cook-Rcp	Debugger displays incorrect values for enumeration types
320545-Voya-Jim	Debugger doesn't work well with Definition



Table A-2 Not Reproducible in D_12_7_3 (continued)

PRS Number	PRS Summary
4348467-Gato-Mboy	Error when private type address is an access type
7178409-Blut-Smp	Cmvc.Show_Image_Of_Generation profile somewhat ignored
7509601-Nati-Drk	Debuggers don't kill themselves properly
7656507-Voya-Phil	Show_Tasks gets exceptions
8757197-Shei-Swb	Problem with Common.Complete
9123690-0140-3	Deleted unit shows up as checked out in CMVC database
9123690-0158-9	CSR5226: Cmvc problem
9123690-0171-1	CSR5649: Cmvc.Import fails with incompatible target keys
9123690-0175-7	CSR6118: Compute_Recoding spews error messages
9123690-0181-1	CSR6122: Anonymous Ada units causes some CMVC operations to hang
9123690-0181-2	CSR6163: Anonymous units cause some CMVC operations to hang
935209-Mago-Trw	Definition does not accept special names



Contents

1. Overview	1
2. Supported Configurations and Upgrades	1
3. Compatibility	2
4. Upgrade Impact	3
4.1. Impact of Specification Changes	4
4.1.1. Upgrading from D_12_6_5 or Later	4
4.1.2. Upgrading from D_12_5_0	4
4.1.3. Upgrading from D_12_2_4	4
4.1.4. Upgrading from D_12_1_1	5
4.2. Impact of Implementation Changes	5
4.2.1. Upgrading from D_12_6_5 or Later	5
4.2.2. Upgrading from D_12_5_0, D_12_2_4, or D_12_1_1	5
5. Known Problems	5
6. New Environment Features	6
6.1. Network Installation	6
6.2. Crash Notification	6
6.3. New Procedures in Package Dfs	6
6.3.1. Reboot_On_Failure	7
6.3.2. Reboot_On_Failure_Interval	7
6.3.3. Quiesce_Reboot_On_Failure	7
6.3.4. Reboot_On_Failure_Settings	7
6.4. New Procedures in System_Maintenance	7
6.4.1. Show_Elaborated_Configuration	7
6.4.2. Show_Session_Of_Job	8
6.5. Rational X Interface Enhancements	8
6.6. New World Network_Public_Archive_Server_Sessions	8
7. Changes from D_12_6_5	8
7.1. Changes to Package Common	9
7.1.1. Selection in Environment Menus	9
7.1.2. Locks and Program_Errors	9
7.2. Changes to Package What	9
7.3. Changes to Archive	10
7.3.1. Archiving into Subsystems	10
7.3.2. Archiving Loaded Main Programs	10
7.3.3. Archiving Searchlists	11
7.3.4. Archiving Ventures	11
7.3.5. Archiving to Tape Across a Network	11
7.3.6. Archive Server and Session Tokens	11
7.4. Compilation Changes	13
7.4.1. Installing Units	13
7.4.2. Coding Units	15
7.4.3. Loading and Executing Units	15
7.4.4. Incremental Compilation	15
7.5. Miscellaneous Library Management Changes	16
7.6. Changes to Crash Analysis	16
7.6.1. Failure Reboot	16
7.6.2. Automatic Notification	17
7.7. Miscellaneous Changes to System Management	18
7.8. Changes to CMVC	18
7.8.1. Make_Controlled and the State Directory	18



7.8.2. Destroy_View and Frozen Objects	18
7.8.3. Naming Conflicts when Copying or Building Views	19
7.8.4. Default Parameter Values for Make_Spec_View	19
7.8.5. Consecutive Calls to Release	19
7.8.6. Cmvc_Access_Control	19
7.8.7. Log Messages	20
7.8.8. Work Orders	20
7.9. Changes to Activity_Implementation	20
7.10. Networking Changes	20
7.11. LRM Interface Changes	21
7.11.1. Discriminated Type Function Changes	21
7.11.2. New Aggregate Range Function	22
7.11.3. Declarations.Kind Change	22
7.11.4. New Associations Package	22
7.11.5. Lexical Typing of Iterators	23
7.11.6. Compilation Unit Pragmas	24
7.11.7. Operations on Generic Instantiations	24
7.11.8. Functions and Enumeration Literal Renames	25
7.11.9. Constants versus Variables	26
7.11.10. Task Entries versus Subprograms	26
7.11.11. Comments in Declaration Names	26
7.11.12. New Labels Function	26
7.11.13. Limited Private Generic Formal Parameters	26
7.11.14. Change to Renaming Declarations Function	27
7.11.15. Renamed Declarations	27
7.11.16. Subunits Without Bodies	27
7.11.17. Subunits Without Specs	27
7.11.18. Declarations.Is_Initialized Change	27
7.11.19. New Function for Incomplete Types	28
7.11.20. Implementation-Dependent Attributes and Pragmas	28
7.11.21. Return Type of Function Instantiations	30
7.11.22. Block and Loop Names	30
7.11.23. Enumeration Literal Queries	30
7.11.24. Record Discriminant and Component Declarations	30
8. Documentation	31
8.1. Printed Documentation	31
8.1.1. Correction to <i>Quick Reference for Parameter-Value Conventions</i>	31
8.2. Online Documentation	31
9. Training	32
A Problem Reports Closed in D_12_7_3	33



*#####
*#####
*#####
*#####

TITLE: r1000_to_apex
TIME PRINTED: Fri Mar 23 11:10:38 2001
TIME QUEUED: Fri Mar 23 11:10:37 2001
PRINTED AT: hp@hplaser (hplj-4) @ mortensen
SUBMITTED BY: elh
DELIVER TO: =====> elh <=====

FLAG VALUES:
a=0, b=0, d=a, f=, g=1, h=, i=0, j=1, l=64, p=10, q=300, s=courier, t=0, u=1,
v=6, w=77, x=2, z=!, A=1, B=gn, C=!, H=, I=, J=+, L=+, N=1, O=3,
P=hp_pcl:hp@hplaser, Q=4, X=ISO8859-1, Y=0, Z=+



```

#set -xv
#!/bin/ksh
#=====
#
#                               Copyright (C) 1994 Celera, Sweden
#=====
==
#
# UNIT NAME:
#@(#) r1000_to_apex Ver 1.9
#
# USAGE:
#     r1000_to_apex model history view|configuration
#
# DESCRIPTION:
#     A post-process to the r1000 command save_set_to_apex_exe
#     this procedure will make real views and subsystems from the
#     files transfered with this command.
#
# OPTIONS:
#     No options
#
# FILES CREATED:
#     The files internal to the views and the subsystems
#
# REVISION HISTORY:
#
#@(#)  Date      Who      Ver      Comment
#@(#)  -----   -
--
#@(#)  94010     msk      1.9     Introdcing the model parameter
#@(#)  941011    lrm      2.0     Rewritten. Removed /csc/ci dependency.
#
#=====
==

Usage="Usage: r1000_to_apex model history view|configuration"

# There should be three parameters
#
if (( $# != 3 )) ; then
    print $Usage
    exit 2
fi

# Apex must be running (this must be run from an Apex shell)
#
if test -z "$APEX_HOME"
then
    print $Usage
    print "This command must be started from an Apex shell"
fi

# Get the parameters
#
model=$1
history=$2
views=$3

# A specified view or a configuration file?
#
if test "`basename $views .cfg`" = "`basename $views`"

```



```

then
    single_view="true"
else
    single_view="false"
fi

# Maintain the subsystem(s)
#
if test "$single_view" = "true"
then
    subsys=${views%.ss*}.ss
    maintain -all $subsys
else
    for view in `cat $views`
    do
        subsys=${view%.ss*}.ss
        maintain -all $subsys
    done
fi

# Remodel all views to the given model
#
remodel -force -model $model $views

# Create target directories and maintain the views
#
if test -n "$APEX_RCI_HOME"
then
    if test "$single_view" = "true"
    then
        control -new_history $history -create_history $views
        crtlib -d -D $views/.local/ada6000 $views
        maintain -all $views
    else
        for view in `cat $views`
        do
            control -new_history $history -create_history $view
            crtlib -d -D $view/.local/ada6000 $view
            maintain -all $view
        done
    fi
fi

```




```
#!/bin/ksh
```

```
for ff in installed.* ; do
```

```
● /bin/cp $ff $ff.old
```

```
/bin/sed 's-^/usr/local/lpp/celera-/opt/celera-' $ff.old > $ff
```

```
/bin/rm -f $ff.old
```

```
done
```

~~Return~~

Run me - at - terminal



tell_about_steps

```
-----
# Run ranlib on .a files for SunOS and AIX only.
#-----
eval ranlib_done=\$ranlib_on_$ARCH
if [ -z "$ranlib_done" ]
then
  if [ -f $prod_home/.asis_version ]
  then
    _v=`$CAT $prod_home/.asis_version`
    asis_a="$prod_base/ada/asis.ss/*$ARCH_OS*.$_v.rel/*.a"
  fi

  if [ -f $prod_home/.asis95_version ]
  then
    _v=`$CAT $prod_home/.asis95_version`
    asis95_a="$prod_base/ada/asis.ada95.ss/*$ARCH_OS*.$_v.rel/*.a"
  fi

  if [ "$ARCH" = sun4 ]
  then
    run_ranlib sun4 Sparc $prod_home/$ARCH/lib/*.a \
      $prod_base/**/*.ss/*$ARCH_OS*.$product_version.rel/*.a \
      $asis_a $asis95_a

    eval ranlib_on_sun4=yes
    save_defaults ranlib_on_sun4
  fi

  if [ "$ARCH" = rs6k ]
  then
    run_ranlib rs6k AIX $prod_home/$ARCH/lib/*.a \
      $prod_base/**/*.ss/*$ARCH_OS*.$product_version.rel/*.a \
      $asis_a $asis95_a

    eval ranlib_on_rs6k=yes
    save_defaults ranlib_on_rs6k
  fi
fi

#-----
# Java setup.
#-----
java_model=$prod_base/c++/model.ss/$ARCH.$product_version.java.rel
java_switches=$java_model/Policy/Switches
if [ -f $java_switches ]
then
  java_home=`$GREP "JAVA_HOME:" $java_switches | $AWK -F: '{print $2}'`
  fmt_mesg "You installed the $product_name Java model.
           This requires the Java compiler to be installed.
           Please provide the directory where this is installed."

  if [ -f $java_home/bin/java ]
  then
    def_dir=$java_home
  else
    unset def_dir
  fi
fi

while true
do
  get_input "Enter the Java compiler directory:" $def_dir
```



```

if [ ! -d $result ]
then
    echo "$result not found or not a directory."
elif [ ! -f $result/bin/java ]
then
    echo "$result/bin/java not found."
    _dir=`$DIRNAME $result`
    if [ -f $_dir/bin/java ]
    then
        echo "However, $_dir/bin/java found."
        result=$_dir
        break
    fi
    echo "$_dir/bin/java not found."
    echo "Are you sure $result is the Java compiler directory?"
else
    break
fi

get_yn "Do you wish to try again?" yes
if [ "$result" = no ]
then
    _line=`$GREP -n "JAVA_HOME:" $java_switches`
    n=`echo $_line | $AWK -F: '{print $1}'`

    warning "Until the correct Java compiler directory is set in
        $java_switches
        you will not be able to compile Java programs using $product_
name.

        Fix line $n in the Switches file,
        $java_switches:"

    $GREP "JAVA_HOME:" $java_switches
    press_cr
    result=$java_home
    break
fi
done

$SED -e "s?JAVA_HOME:.*?JAVA_HOME: $result?" \
    $java_switches > /tmp/apex.sh.$$

if diff $java_switches /tmp/apex.sh.$$
then
    echo
    echo Up-to-date: $java_switches
else
    if $CHMOD u+w $java_switches
    then
        $CP /tmp/apex.sh.$$ $java_switches
        $RM -f /tmp/apex.sh.$$
        echo
        echo Updated: $java_switches
    fi
fi
fi

#-----
# Summit setup.
#-----
if [ -f $prod_home/.enable_task ]
then
    if [ -s $rational_dir/config/task_domain_path ]
    then

```



```

for _dir in `sed -e "s?:? ?g" $rational_dir/config/task_domain_path`
do
    def_domain_dir=${def_domain_dir:-$_dir}
    domain_dirs="$domain_dirs
$_dir"
done
unset _dir

task_ss=`$DIRNAME $def_domain_dir`
task_name=`$BASENAME $task_ss .ss`
task_dir=`$DIRNAME $task_ss`
else
    task_name=`echo "${SV_COMPANY_NAME:-tasks}" |
        $SED -e "s~[!@#%&*()-+=|:;<>,./]~~g" -e "s~ ~~~g"`
    task_dir=$prod base/task
    task_ss=$task_dir/$task_name.ss
fi

while true
do
    fmt_mesg "You have installed Summit. Please select a
    directory to put your company's task management subsystem in."
    get_input "Input directory:" $task_dir
    if [ -d $result ]
    then
        if [ -w $result ]
        then
            echo "Okay, $result exists and you can write in it."
            break
        else
            echo "You do not have write permission on"
            echo "  $result."
            echo "Enter a directory that you can write in."
        fi
    elif mkdir_p $result
    then
        echo "Okay, $result created."
        break
    else
        echo "Cannot create $result."
    fi
done
task_dir=$result

get_input "Enter a name for your Summit subsystem:" "$task_name"
task_name=`$BASENAME $result .ss`
task_ss=$task_dir/$task_name.ss
task_domain=$task_ss/all.wrkr

if [ -n "$def_domain_dir" ]
then
    if [ "$def_domain_dir" != "$task_domain" ]
    then
        _mesg="Select the default domain:"
        _type="default domain"
        select_misc $domain_dirs $task_domain
        echo
        echo "Updating $rational_dir/config/task_domain_path"
        # The first (result) in the list is the default.
        echo $result > $rational_dir/config/task_domain_path
        for _dir in $domain_dirs $task_domain
        do
            if [ "$_dir" != "$result" ]
            then

```




```

        echo $_dir
    fi
done >> $rational_dir/config/task_domain_path
task_domain=$result
save_defaults task_domain
else
    if [ -d $def_domain_dir ]
    then
        unset task_ss
    fi
    echo
    echo Up-to-date: $rational_dir/config/task_domain_path
fi
else
    echo
    echo "Saving: $task_domain"
    echo "    in: $rational_dir/config/task_domain_path"
    echo $task_domain > $rational_dir/config/task_domain_path
    save_defaults task_domain
fi

# Set ownership and permissions for company's task management subsystem.
if [ -n "$task_ss" ]
then
    $RM -f user_group_file.$$
    $TOUCH user_group_file.$$
    user=`$LSL user_group_file.$$ | $AWK '{print $3}'`
    group=`$LSL user_group_file.$$ | $AWK '{print $4}'`
    $RM -f user_group_file.$$

    fmt_mesg "By default, your task management subsystem will be owned b
        $user in group $group. You can select another user to own it,
        but if you do you will need to run the '$root_shell' script as
        root to complete the install."

    get_yn "Use '$user' as the owner id for $task_name.ss" yes
    if [ $result = no ]
    then
        get_input "Enter the user id for the owner of $task_name.ss"
        user=$result
        get_input "Enter the group id to use for $task_name.ss"
        group="$result"
        need_root=yes
    else
        need_root=no
    fi

    _mesg="Select the permissions that $task_name.ss will have:"
    _type="permissions"
    _def="$task_permissions"
    select_misc "Owner ($user) and group ($group) updateable." \
        "Updateable by anyone."
    task_permissions="$result"
    save_defaults task_permissions
    case "$task_permissions" in
        Owner* )
            task_ss_chmod="chmod -R g+w $task_ss"
            ;;
        * )
            task_ss_chmod="chmod -R a+w $task_ss"
            ;;
    esac

```



```

if [ "$need_root" = yes ]
then
    echo "#!/bin/sh -x" > $root_shell
    echo "chown -R $user $task_ss" >> $root_shell
    echo "chgrp -R $group $task_ss" >> $root_shell
    echo "$task_ss chmod && $RM -f $root_shell" >> $root_shell
    $CHMOD 777 $root_shell
    unset task_ss_chmod
fi
fi
fi

#-----
# Fix the Spire.cfg file for the C++ compiler.
#-----
eval spire_cfg=$prod_base/\${ARCH}_spire_cfg
if [ "$ARCH" = sun4 -a -f "$spire_cfg" ]
then
    def_dir=`$GREP "^SUN-LIBDIR:INFO:" $spire_cfg | $AWK -F: '{print $3}'`
    fmt_mesg "You installed the $product_name C++ product.
              This requires the SunOS C++ compiler library to be installed.
              Please provide the directory where this is installed."

    while true
    do
        get_input "Enter the C++ compiler library directory:" $def_dir
        if [ ! -d $result ]
        then
            echo "$result not found or not a directory."
        elif [ ! -f $result/crt0.o ]
        then
            echo "$result/crt0.o not found."
            echo "Are you sure $result is the C++ compiler library directory?"
        else
            break
        fi

        get_yn "Do you wish to try again?" yes
        if [ "$result" = no ]
        then
            _line=`$GREP -n "^SUN-LIBDIR:INFO:" $spire_cfg`
            n=`echo $_line | $AWK -F: '{print $1}'`

            warning "Until the correct C++ library is set in
                    $spire_cfg
                    you may have trouble linking some C++ programs using $product
_name.

                    Fix line $n in the Spire.cfg file,
                    $spire_cfg:"

            $GREP "^SUN-LIBDIR:INFO:" $spire_cfg
            press_cr
            result=$def_dir
            break
        fi
    done

    $SED -e "s?^SUN-LIBDIR:INFO:.*?SUN-LIBDIR:INFO:$result:?" \
        $spire_cfg > /tmp/apex.sh.$$

    if diff $spire_cfg /tmp/apex.sh.$$
    then
        echo
    fi

```



```

        echo Up-to-date: $spire_cfg
    else
        if $CHMOD u+w $spire_cfg
        then
            $CP /tmp/apex.sh.$$ $spire_cfg
            $RM -f /tmp/apex.sh.$$
            $CHMOD u-w $spire_cfg
            echo
            echo Updated: $spire_cfg
        fi
    fi
fi

#-----
# Set APEX_LOCKING before patching apexinit.
#-----

fmt_msg "Apex can automatically recover abandoned file locks that were
acquired by a process no longer running another machine. To do this,
Apex must be able to run a 'remote' command that checks process
existence on that other machine. There are several options for how
this can be done:"

echo "
Rsh:      use the ``$BASENAME $RSH`` remote shell command (recommended)
No_Remote: don't attempt such lock recoveries
On:       use the 'on' command (only for SunOS and AIX)"

case $ARCH in
    sun4|rs6k )
        _def=On
        ;;
    * )
        _def=Rsh
        ;;
esac

fmt_msg "For further information about the APEX_LOCKING switch, refer
to the Installation Guide."

_msg="What form of remote process existence testing should be
the default for Apex users?"
_type=option
select _misc Rsh No_Remote On
APEX_LOCKING="Remote=>$_result"

#-----
# Patch apexinit.sh
#-----
add_cleanup_file /tmp/patch_script.$$
echo "APEX_BASE=$APEX_BASE" > /tmp/patch_script.$$
echo "APEX_HOME=$prod_home" >> /tmp/patch_script.$$
echo "APEX_LICENSE_FILE=$SV_LIC_FILE_NAME" >> /tmp/patch_script.$$
echo APEX_LOCKING="\$APEX_LOCKING\" >> /tmp/patch_script.$$

./share/patch_script \
    /tmp/patch_script.$$ \
    $prod_home/bin \
    $prod_home/bin/apexinit.sh

#-----
# Make the Frame 'links' file and the 'apex_menus' for on-line documents.
#-----
if [ -x ./share/make_links_file ]

```



```

then
    ./share/make_links_file
fi
if [ -x ./share/make_apex_menus ]
then
    ./share/make_apex_menus
fi

#-----
# Make on-line doc area read only.
#-----
if [ -d $prod_home/doc/En_US ]
then
    echo
    echo "Making $prod_home/doc/En_US read-only..."
    $CHMOD -R a-w $prod_home/doc/En_US
fi

#-----
# Create links to Apex_Main
#-----
$BIN=$prod_home/$ARCH/bin
echo
echo "Creating links to Apex_Main"
echo "in $BIN..."

if [ -x $BIN/Apex_Main_Duo ]
then
    $RM -f $BIN/Apex_Main
    $LN $BIN/Apex_Main_Duo $BIN/Apex_Main
elif [ -x $BIN/Apex_Duo_Main ]
then
    $RM -f $BIN/Apex_Main
    $LN $BIN/Apex_Duo_Main $BIN/Apex_Main
elif [ -x $BIN/Apex_Main_Ada ]
then
    $RM -f $BIN/Apex_Main
    $LN $BIN/Apex_Main_Ada $BIN/Apex_Main
elif [ -x $BIN/Apex_Main_Cpp ]
then
    $RM -f $BIN/Apex_Main
    $LN $BIN/Apex_Main_Cpp $BIN/Apex_Main
fi

if [ -f $BIN/Apex_Main_Duo ]
then
    $RM -f $BIN/Apex_Main_Ada $BIN/Apex_Main_Cpp
elif [ -f $BIN/Apex_Main_Ada ]
then
    $RM -f $BIN/Apex_Main_Cpp
fi

if [ ! -f $prod_home/.enable_task ]
then
    ccl_opt="-notsk"
fi

if [ ! -f $prod_home/.enable_apex_cpp ]
then
    ccl_opt="$ccl_opt -nocpp"
fi

if [ ! -f $prod_home/.enable_apex_ada83 -a ! -f $prod_home/.enable_apex_ada95

```




```

]
then
    ccl_opt="$ccl_opt -noada"
    $RM -f $BIN/Apex_Ada_Main
fi

# Create links.                                     # ignore warnings.
echo $BIN/Apex_Main Create_Command_Links $BIN $ccl_opt | $GREP -v " !!! "
$BIN/Apex_Main Create_Command_Links $BIN $ccl_opt

if [ -x $BIN/Apex_Ada_Main ]
then
    echo
    echo "Creating links to Apex_Ada_Main too..."          #ignore warnin
gs
    echo $BIN/Apex_Main Create_Command_Links $BIN -replace -adamain
    $BIN/Apex_Main Create_Command_Links $BIN -replace -adamain | $GREP -v " !
!! "
fi

if [ -d $prod_home/editor_files -a ! -d $prod_home/editor_files/null.session
]
then
    $MKDIR $prod_home/editor_files/null.session
fi

#-----
# Setup EMACS lock directory.
#-----
if [ -f $APEX_HOME/.emacs_version ]
then
    emacs_version=`$CAT $APEX_HOME/.emacs_version`
    EMACS_HOME=$APEX_BASE/cots/emacs-apex.$emacs_version/$APEX_ARCH

    if [ ! -d $APEX_BASE/cots/emacs/lock ]
    then
        mkdir_p $APEX_BASE/cots/emacs/lock
        $CHMOD 777 $APEX_BASE/cots/emacs/lock
    fi

    if [ -d $EMACS_HOME/lib/emacs ]
    then
        $RM -f $EMACS_HOME/lib/emacs/lock
        $LN -s $APEX_BASE/cots/emacs/lock $EMACS_HOME/lib/emacs/lock
    fi
fi

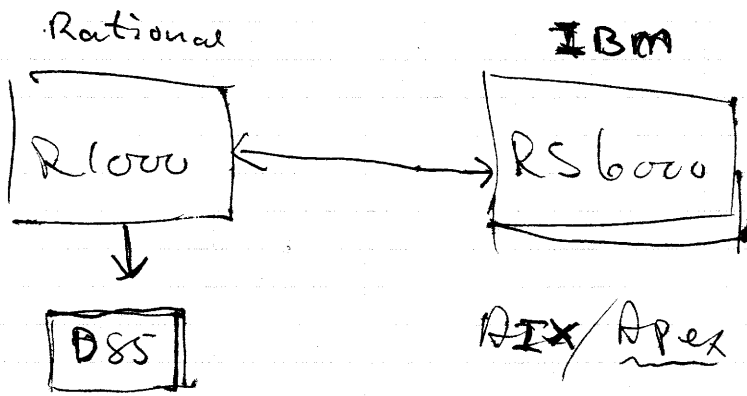
#-----
# Create startup file for apexinit
#-----
$TOUCH /tmp/lp.start.$$
add_cleanup_file /tmp/lp.start.$$

```



6/3 2001

$\frac{10^{15}}{12^{02}}$



10 ADA status
kan 3 Apex

Rational
R1000 spareparts.

key: Config styring / Historik.

skal kunne generere alle tidligere releases.

Rational AOA til NT?

Værktøjer på Rational? ~~hvad~~ Hvad er der, hvad skal vi bruge?

Primo April Status

1021

1021
1021
1021

1021
1021

1021
1021

1021
1021

1021
1021

1021
1021

1021
1021

1 Omfordeling af Rational belastning

Rational udviklingsmiljøet har igennem en længere periode givet anledning til bekymring på flere fronter, bl.a.:

<u>Driftssikkerhed:</u>	Reserve dele til Rationalerne er svære/umulige at få fat i. Det kan derfor få store konsekvenser, hvis en Rational bryder ned.
<u>Svartider:</u>	De fleste udviklere føler at de spilder megen tid, fordi Rationalen er for langsom. Derudover er der perioder, hvor Rationalen lever sit eget liv i stedet for at gøre hvad brugerne ønsker.
<u>Psykologisk:</u>	Rationalen er en ældgammel konstruktion, og derfor ikke så spændende som en moderne computer.

Formålet med dette dokument er at beskrive nogle mekanismer til at begrænse belastningen på vores Rationaler. Mekanismerne kan deles op i følgende områder:

- Omfordeling mellem Rationalerne
- Bedre kendskab til Rationalens virkemåde
- Etablering af APEX udviklingsmiljø

Den følgende beskrivelse vil gennemgå hver af disse områder i detaljer.

1.1 Omfordeling mellem Rationalerne

Rationalerne anvendes i øjeblikket ikke optimalt. Diskene er overfyldt, samtidig med at stort set de samme ting ligger på alle 3 Rationaler. Det er også et mål om muligt at begrænse anvendelsen til 2 Rationaler i daglig drift, hvor den 3. Rational så kan stå standby som reserve.

Efter omfordeling kunne Rationalerne f.eks. anvendes som følger:

Rat I (integration): På integrationsmaskinen samles alle 4 system software konfigurationer (MU, MCM, CV og IS). Kun een konfiguration ad gangen bør være kompileret og linket. I0_Working views er fælles for alle konfigurationer, således at en version inkluderet til en release automatisk bliver anvendt til de følgende releaser. Integrationsmaskinen skal ikke indeholde Rat views og udviklings views.

Rat U (udvikling): På udviklingsmaskinen samles alle udviklings views. I0_Working views er fælles for al udvikling og skal opdateres når et view releases til integration. En Rat testbed skal bibeholdes for Ada uddannelse, hvorimod Rat_Working views kun bibeholdes hvis disse er holdt opdateret. Udviklingsmaskinen skal ikke indeholde system software, men et mindre view til udvikling og test af MMI grafik kan etableres, når det er nødvendigt.

Rat X (standby): Reservemaskine. Diskene kan evt. anvendes til at have en kopi af den senest frigivne release.

1.1.1 Vejen frem til separate integrations og udviklings maskiner

Følgende trin skal gennemgås for at etablere den nødvendige funktionalitet på de to maskiner:



1. Forberedelse af Rat 1 som integrationsmaskine. Udviklings views flyttes til Rat 3. Rat testbed og Rat udviklings views køres ud på bånd. Rat udviklings views og testbed slettes.
2. MU system software demotes. Imports til MU system software registreres og fjernes. IO_Working views på Rat 1 opdateres med seneste releasede versioner (fundament til MU).
3. Imports til MU system software genetableres. MU kompiles og linkes.
4. MCM og IS system software fra Rat 3 kopieres til Rat 1. Imports registreres og fjernes.
5. CV system software fra Rat 2 kopieres til Rat 1. Imports til CV registreres og fjernes.
6. Forberedelse af Rat 3 som udviklingsmaskine. System software demotes og slettes. Alle Rat udviklings views køres ud på bånd. Rat udviklings views slettes. Rat testbed bibeholdes.
7. IO_Working views på Rat 3 opdateres med seneste releasede versioner (anvend shared_closure_only). IO_Working views, der kun indeholder programmer, slettes. IO_Working views kompiles.
8. Udviklings views fra Rat 2 kopieres til Rat 3. Kompilering efter behov.
9. Forberedelse af Rat 2 som reserve maskine. Rat 2 lukkes ned.

Trin 1-3 er en naturlig del af integrationsarbejdet for MU. Trin 4-9 kan foregå parallelt med SWAT test for MU.

1.2 Bedre kendskab til Rationalens virkemåde

Udviklerne bør bibringes et bedre kendskab til Rationalens virkemåde og faciliteter, således at man bedre forstår hvad man selv kan gøre for at forbedre sine svartider.

Heri skal bl.a. medtages centrale emner som:

- Inkrementel opdatering
- Kompilering som baggrundsjob
- Snapshots
- Garbage Collection
- System backup

1.3 Etablering af APEX udviklingsmiljø

APEX kan etableres som en aflastning for Rationalerne til større udviklingsopgaver. Til mindre opdateringer vil det hurtigste stadigvæk være at anvende Rationalen, da man så undgår overheadet med kopiering frem og tilbage mellem Rational og APEX.

CDF til APEX vil selvfølgelig være den bedste løsning på lang sigt, således at al udvikling og funktionel test kunne udføres herfra. Men indtil dette er muligt må vi basere os på at kopiere koden fra APEX til Rational før aftenstning. Værktøjet til kopiering af udviklingsviews fra APEX til Rational (Delta) er i denne forbindelse yderst ønskværdigt.

APEX udviklingsmiljøet kan etableres og udvides uafhængigt af omfordelingen af Rationalerne. En ekstra APEX maskine med tilhørende licenser skal kobles fast på Ethernet til Rationalerne. Den eksisterende APEX maskine bibeholdes til MMI generering.

APEX maskinen kan anvendes som backup for releasede system funktioner og system software.



*** AFSENDELSES RAPPORT ***

AFSENDELSE OK

AF/MT-NR 4384
FORBINDELSE TLF 43351001
UNDERADRESSE
FORBINDELSE ID
STARTTID 09/08 14:12
BRUGSTID 02'43
ARK 4
RESULTAT OK

Fax

TERMA[®]

Fax 43351001
Firma Rational Software A/S
Att David Jonsson
Fra Erlo Haugen
Dato 09. august 2001
Side/r 4, incl. denne
Reference
Emne E-mails fra Greg Bek

Terma A/S
Naval & Communications
Systems Division
Mårkærvej 2
DK-2630 Tåstrup
Denmark
T +45 4352 1513
F +45 4352 5758
terma.ncs@terma.com
www.terma.com
CVR/VAT No.: 41881828
(Aarhus)
Bankers: Danske Bank

Jeg sender lige de mails jeg har vedr. sammenkobling R1000<->Apex. Jeg håber at det er den rette mand vi har fat i (Greg Bek)

Venlig hilsen


Erlo Haugen



Fax

TERMA[®]

Fax 43351001
Firma Rational Software A/S
Att David Jonsson
Fra Erlo Haugen
Dato 09. august 2001
Side/r 4, incl. denne
Reference
Emne E-mails fra Greg Bek

Terma A/S
Naval & Communications
Systems Division
Mårkærvej 2
DK-2630 Tåstrup
Denmark
T +45 4352 1513
F +45 4352 5758
terma.ncs@terma.com
www.terma.com
CVR/VAT No.: 41881828
(Aarhus)
Bankers: Danske Bank

Jeg sender lige de mails jeg har vedr. sammenkobling R1000<->Apex. Jeg håber at det er den rette mand vi har fat i (Greg Bek)

Venlig hilsen


Erlo Haugen

Vigtigt

Denne fax er personlig og er derfor beskyttet af brevhemmelighed. Hvis du har modtaget denne fax ved en fejtagelse kontakt os da venligst med det samme på telefon eller fax.



Michael Jørgensen

From: Tove Rosendahl
Sent: 9. august 2000 10:28
To: Dan Hauge
Cc: Erling Carstensen
Subject: FW: R1000 conversion

Hej Dan! (og Erling)

Jeg har fået nedenstående fra Rational. Endelig er der nogen som er vågnet op, og ser på vort problem. Jeg har sendt dette til ERC også, idet jeg håber, han kan få tid til at læse og kommentere alternativ 2.

Mvh Tove

-----Original Message-----

From: Christian Juul [SMTP:cjuul@rational.com]
Sent: 9. august 2000 09:59
To: tor@terma.com
Subject: R1000 conversion

Tove Rosendahl

Vore teknikere har internt drøftet mulighederne for at aflaste jeres R1000 miljø. Der er, efter min opfattelse, en del intern uenighed hos Rational omkring de tekniske muligheder.

Jeg vil dog her præsentere jer for de 2 alternativer som en af vore teknikere, Greg Bek ser.

Alternative 1. Buy/License CDF 9 or 10 from CelsiusTech.

- >
- > This has many advantages:
- > - No more need for R1000's
- > - Off the shelf IBM hardware (I could be wrong if CDF9/10
- > only run on old versions of the AIX OS)
- > - Much faster build times
- > - Much easier backup/restore
- > - reduced system management overhead
- > - Network based development
- >
- > Of course it has disadvantages:
- > - Dealing with Celsius to get the CDF
- > - System retest and rebuild with new compiler will be
- > a huge job

> I think this is the best long term solution.

> Alternative 2. Mixed Delta/Apex

> Alternative 2.1 CM still done on Delta

- >
- > This would involve building an RCI customization on Delta
- > to drive Apex on UNIX. This has never been done, but should
- > only take about 2 - 3 days for a competent Tech Rep or Customer
- > Toolsmith. Of course finding the Rational TR who knows anything
- > about Delta is a challenge (and I don't want to visit Denmark
- > that desparately). The RCI customization wouldn't need to
- > have all the compilation management implemented as the analyze
- > etc. would take place within Apex as part of normal development.
- >
- > The RCI has a command called: Rci.Accept_Remote_Changes, this
- > command finds changed units on the remote compilation host (Apex
- > on UNIX in this case) and transfers the changed files back to
- > the R1000. The command can be set to do actual file comparisons
- > (slow) or just time stamps. The command automatically checks
- > out the unit on the R1000 and checks-in the changed file to
- > maintain version history.

> After Accept_Remote_Changes was run then CMVC.Accept_Changes
> (if I can still remember my commands correctly) would need
> to be run to update the CDF views with the new versions so
> that a build could then be run.



- >
- > Alternative 2.2 CM Done with Apex
- > This would involve having the same RCI customization as in 2.1,
- > enough said.
- >
- > Before starting however all the CM data would be migrated to
- > Apex using the migration tool on Delta. This tool does exist
- > (I know, as I've used it and modified it in the past).
- >
- > Then RCI.Accept_Remote_Changes could be used to transfer changed
- > units back for compilation on the R1000, but the units would
- > be uncontrolled on the R1000 so the check-out/check-in step
- > would be avoided.
- >
- > The CMVC.Accept_Changes would still be needed.
- >
- > This alternative has the advantage of using the much faster
- > CMVC engine on UNIX, as well as easier backup etc. This would
- > also ease any future transition to a CDF 9/10 solution.
- >
- >
- > One of the big risks Terma face is a very rapid decline in
- > R1000 skills within Rational and their own organization.
- >
- > Greg

Jeg vil foreslå vi drøfter dette nærmere, når I har vurderet de 2 alternativer.

Hvis du har spørgsmål eller kommentarer er du naturligvis meget velkommen til at kontakte mig.

Med venlig hilsen
Rational Software

Christian Juul
Tlf. 43 35 10 00
Mobil 40 15 57 65



Michael Jørgensen

From: Christian Juul [cjuul@Rational.Com]
Sent: 6. juli 2000 09:53
To: dah@terma.com
Subject: R1000 til Apex

Hej Dan

Har måttet udsætte min ferie til imorgen, og jeg er derfor først tilbage medio uge 30.

Jeg har fået en interessant mail fra Greg Bek i US, som I kan vurdere nærmere.

The interoperability of Delta and Apex was always assumed that a migration from Delta to Apex was needed.

We never designed from transferring back from Apex to Delta.

It could be done, of course, but it would be some work. Basically the Delta box would have to be setup to use the RCI so that the Rci.Accept_Target_Changes command could be run to automate the process of bringing changes made in Apex back into Delta.

Alternatively you could do this with a program on Delta that ran FTP to get the source files and then parsed those changes back into the appropriate subsystems and views in Delta. Intelligent use of file compare would help.

Using Apex for development and unit testing and Delta for integration would take a load of the R1000's.

Greg

Med venlig hilsen
Rational Software

Christian Juul

Tlf. 43 35 10 00
Mobil 40 15 57 65

